IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 4 December 2023, accepted 5 January 2024, date of publication 11 January 2024,
date of current version 19 January 2024.

*Digital Object Identifier 10.1109/ACCESS.2024.3352629*

RESEARCH ARTICLE

# DEPHIDES: Deep Learning Based Phishing Detection System

**OZGUR KORAY SAHINGOZ[1], EBUBEKIR BUBER[2], AND EMIN KUGU[3], (Member, IEEE)**
[1]Computer Engineering Department, Biruni University, 34100 İstanbul, Turkey
[2]Computer Engineering Department, Yıldız Technical University, 34469 İstanbul, Turkey
[3]Software Engineering Department, TED University, 06420 Ankara, Turkey

Corresponding author: Emin Kugu (emin.kugu@tedu.edu.tr)

**ABSTRACT** In today's digital landscape, the increasing prevalence of internet-connected devices, including smartphones, personal computers, and IoT devices, has enabled users to perform a wide range of daily activities such as shopping, banking, and communication in the online world. However, cybercriminals are capitalizing on the Internet's anonymity and the ease of conducting cyberattacks. Phishing attacks have become a popular method for acquiring sensitive user information, including passwords, bank account details, social security numbers and more, often through social engineering and messaging tools. To protect users from such threats, it is essential to establish sophisticated phishing detection systems on computing devices. Many of these systems leverage machine learning techniques for accurate classification. In recent years, deep learning algorithms have gained prominence, especially when dealing with large datasets. This study presents the development of a phishing detection system based on deep learning, employing five different algorithms: artificial neural networks, convolutional neural networks, recurrent neural networks, bidirectional recurrent neural networks, and attention networks. The system primarily focuses on the fast classification of web pages using URLs. To assess the system's performance, a relatively extensive dataset of labeled URLs, comprising approximately five million records, was collected and shared. The experimental results indicate that convolutional neural networks achieved the highest performance, boasting a detection accuracy of 98.74% for phishing attacks. This research underscores the effectiveness of deep learning algorithms, particularly in enhancing cybersecurity in the face of evolving cyber threats.

**INDEX TERMS** Deep learning, cyber security, phishing attack, classification algorithms, phishing detection.

## I. INTRODUCTION

Today, the internet is used to carry out and manage a large portion of critical activities. All sectors, such as shopping and banking, which have intense financial activities, have been serviced over the internet. In addition to increasing the availability of companies, this situation also brings many security vulnerabilities. Due to the uncontrolled and anonymous nature of the Internet, cyber-attackers can easily perform attacks, resulting in severe damage as a result.

It has been one of the responsibilities of companies to ensure that their employees/customers are not harmed by attacks against them. So, companies are working hard to take the necessary precautions to ensure that their customers are not affected by cyber-attacks. Phishing attacks are the most well-known cyber-attacks aimed at deceiving customers. Phishing is a form of fraud in which the attacker tries to learn sensitive information, such as login credentials or account information, by sending it as a reputable entity or person via email or other communication channels.

Typically, a victim receives a message that appears to have been sent by a known contact or organization. The message contains malicious software targeting the user's computer or has links to direct victims to malicious websites to trick them into divulging personal and financial information, such as passwords, account IDs, or credit card details. Attackers use some specific digital tricks to hide themselves and deceive users without being noticed. This makes it quite easy to deceive users who are not aware of phishing attacks. It is easier to carry out this type of attack that exploits people's

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

weaknesses than other types of attacks. Because the main elements needed for the initiation of the attack are based only on the creation of content that the users will believe.

Due to their widespread use, phishing attacks are the focus of many cyber security researchers and academic groups. The Anti-Phishing Working Group (APWG) [1] has had a major impact on the research of phishing attacks, and they emphasize that most of the phishing attacks target the victim to gather their sensitive information by acting as if they were legal sites to get sensitive information.

According to the APWG Phishing Activity Trends Report in the second quarter of 2022, the Unique Phishing attack were increased by almost 43% in 12 months and the most targeted industry sectors by phishing attacks were financial institutions, SAAS/Webmail, social media, and payments [1]. Additionally, according to Cloudflare's 2023 Phishing Threats Report these attackers mainly use different tactics whose distribution is shared in Fig. 1, in which the most popular one is use of Deceptive Links (URL Addresses) [2].

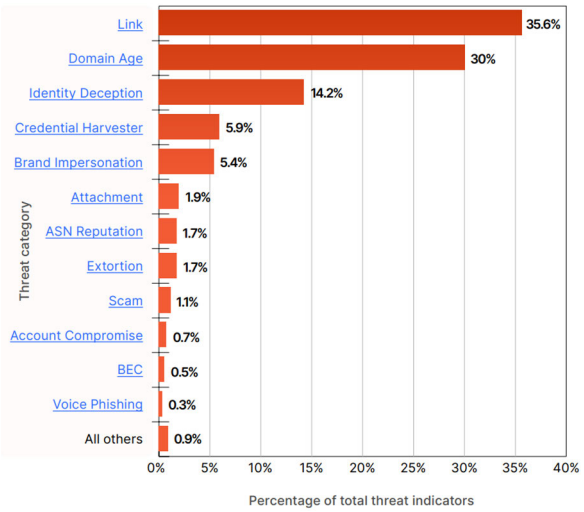


**FIGURE 1. Phishing statistics in the second quarter of 2022.**

Apart from other cyber-attacks, phishing aims to steal identities by exploiting the vulnerabilities of computer users. Even though many security products have been put in place to keep the network infrastructure safe and the systems are always being watched, companies can lose a lot of money if a user accidentally shares sensitive information.

Phishing attacks are based on referring the user to a web page that looks legal. These web pages seem to belong to a real company or web application. This information is stolen by the attacker when the user enters the credentials in the form on this page. An attacker can use a variety of methods to direct users to the web page they want. The most common way is to send an email to the user and direct them to the malicious web page with the link in the message. Fig. 2 shows what a phishing email looks like and what it looks like on the web. Currently, several sophisticated tools can be used to create a phishing web page with a legal appearance, and even people who do not have technical knowledge can create a phishing web page very quickly. Therefore, this type of attack is quite simple and can be preferred even by novice users. Attackers can perform phishing attacks either by targeting large masses of users or by targeting a specific user segment.

As a result of these properties, there are a growing number of phishing attacks all over the world. Phishing attacks target the weakest part of the security chain, end users. Volkamer et. al. [3] emphasize that computer users may fall into phishing due to the following reasons:

- Limited knowledge about the URLs and their structure,
- Not knowing about trustable web pages,
- Due to the use of hidden URLs and redirections, original website address/URL can be hidden from the users and not displayed in the message,
- Due to the workload of the users, they omit to consult the web pages/URLs, or they can accidentally enter to them,
- Users don't know anything about phishing and cannot distinguish legitimate web pages from phishing ones.

Phishing attacks are a type of cyber-attack based on exploiting the weaknesses of computer users. For this reason, a two-layer security mechanism should be considered to prevent this attack. The first of these layers includes steps to increase user awareness. It should be ensured that the awareness of the users is always high by using factors such as repeated training and warnings. To prevent phishing attacks, the other security layer includes the development of the necessary software to detect and prevent these attacks before they reach the end user.

There are several developed approaches to detection of phishing web pages such as machine learning based trained systems, list-based system which use blacklists/whitelists, image-based systems that take care of visual similarities and third party-based systems which are connected to DNS or whois based web services.

On the other side, in the literature, it is easily seen that there is a growing tendency towards the use of machine learning-based solutions due to their dynamicity and adaptability to new attack types. At the same time, in recent years, a specific learning mechanism, named deep learning, has emerged, and it is especially useful for training big data systems or systems that do not have some defined features and the tendency has shifted to deep learning mechanisms in recent years [4], [5]. In this paper, we aim to implement a real-time detection of phishing web pages by investigating their URL addresses by using a deep learning (DL) based trained system. The developed DL model exhibits a notable capability to identify previously unrecognized phishing attacks. This encompasses scenarios where phishing threats have not been shared with any external source beforehand. Notably, even in instances where users initiate requests to URLs classified as phishing, the model enables preemptive measures by blocking access to these URLs, thereby

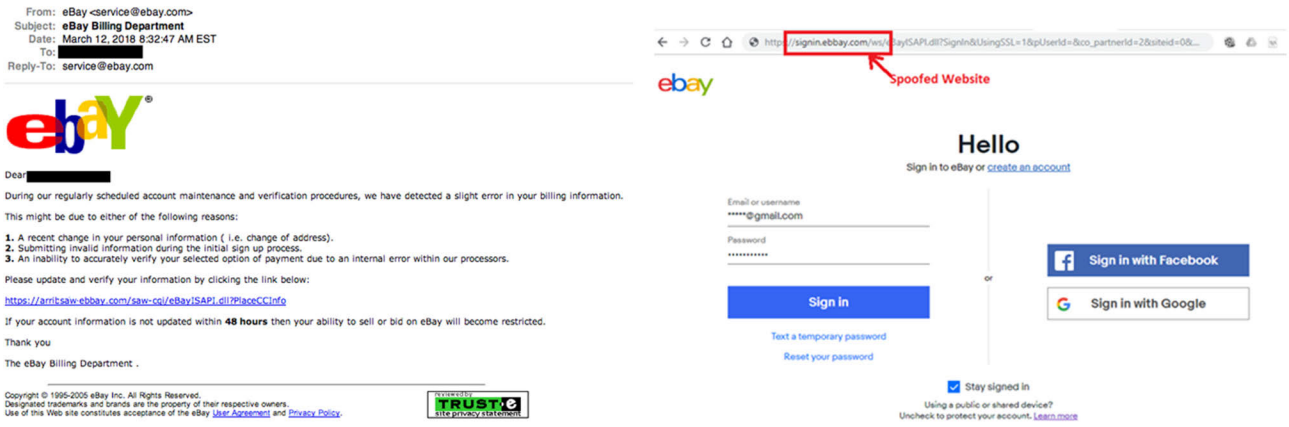


**FIGURE 2.** Sample of a phishing mail and a web page.

preventing potential attacks. This proactive approach ensures that a significant proportion of zero-day phishing attacks can be effectively detected and blocked. The proposed work does not employ third-party solutions, including Whois Data, Web of Trust, Google Safe Browsing, and URL/IP Blacklists. This intentional exclusion facilitated faster detections, emphasizing a self-contained and independent approach in the study's methodology. The results obtained from traditional machine learning methods were analyzed in this article and the results were compared. The experimental and comparative results showed that the proposed solution gives incredibly attractive results especially compared with the standard machine learning based methods. The major contributions of the paper are:

- Construction of a Well-Balanced Large Dataset: The creation of a sizable dataset with a balanced class distribution stands out as a substantial and challenging achievement, particularly when compared with existing datasets.
- Enhanced Phishing Detection with Reduced False Positives: The paper demonstrates improved phishing detection capabilities, showing a reduced false positive rate. This refinement is crucial for enhancing the reliability of the detection system.
- Fast Phishing Page Detection via URL Analysis: The ability to rapidly detect phishing pages by analyzing the URL addresses represents a significant advancement. This quick analysis adds a layer of efficiency to the phishing detection process.
- Highly Accurate Detection of "Zero-Day Attacks": The paper introduces a method for accurately identifying "Zero-Day Attacks," contributing to an increased level of security. This capability is pivotal for staying ahead of emerging threats by distinguishing between normal and abnormal requests.
- Independence from Third-Party Services: The proposed system operates independently of third-party services, reducing reliance on external sources and enhancing the self-sufficiency of the detection mechanism.
- Utilization of Large-Scale Data Analysis for Training: The incorporation of huge size data analysis in training the system with a vast dataset signifies a robust approach. This utilization of extensive datasets enhances the system's learning and adaptability.
- Language-Independent Detection Process: The paper introduces a language-independent detection process, underlining the versatility of the proposed system across different languages. This adaptability is crucial for addressing the diverse nature of phishing attacks.

The rest of the paper is organized as follows: related works about phishing detection are detailed in Section II. Section III focuses on background knowledge about the methods and tools. The details of the proposed system are depicted in Section IV. The experimental results and discussions are presented in Section V and Section VI respectively. Finally, conclusions and future works about the research are presented.

## II. RELATED WORK

Phishing attacks are based on taking advantage of people's weaknesses. Therefore, the first step to ensuring security against phishing attacks is to raise awareness of the users and to keep their awareness high. Through simulations of sample scenarios, it is necessary to assess the awareness levels of the users at regular intervals and repeat the training. Users, even experienced ones, can fall victim to this type of attack. Although security training decreases the number of deceived users, software-based detection systems, which can be categorized mainly into five groups as depicted in Fig. 3, can also warn the user about suspicious web pages. List Based Detection Systems

List-based phishing detection systems are the ones that can be easy to set up and maintain. They mainly use two lists as blacklists and whitelists for classifying URL addresses and/or IP addresses. These addresses can be classified either through voting or by using experienced human classifiers who receive computerized support for their decision.

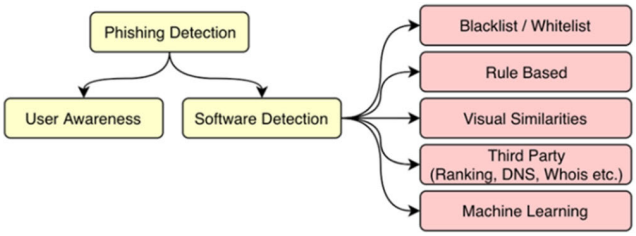


**FIGURE 3.** Phishing detection model.

The Google Safe Browsing API enables the client applications to check the URLs against the constantly updated blacklists [6]. Similarly, Office 365 provides an Advanced Threat Protection service for enterprise organizations, which checks the URLs when the user clicks on them in the email message [7]. The system immediately checks this address before opening it and identifies itself either as safe, malicious or blocked.

On the other hand, Jain and Gupta [8] implemented an auto-updated whitelist-based phishing detection system, which aims for a faster access time with a high detection rate. They tested their proposed system with a limited number of phishing and legitimate datasets (with 1120 phishing and 405 legitimate addresses) and reached an 86.02% true positive rate and a 1.48% false negative rate in their work.

In [9] Azeez et al. demonstrate how to detect phishing attacks using an automated white-list method. The whitelist is developed following an in-depth study of both the visible and genuine connections. The similarities of the known trustworthy site are computed by juxtaposing the domain name with the whitelist's contents, comparing it to the IP address prior to making a judgment, and then assessing the real link and visual link and calculating the known trustworthy site's similarities.

The main advantage of these systems is their quick response time. Therefore, they are suitable for real-time environments. However, it is hard to maintain the up-to-dateness of the lists. Therefore, these systems are vulnerable to zero-day attacks on websites, which have not been used previously.

### A. RULE-BASED DETECTION SYSTEMS

Depending on the features of the web pages, these types of systems detect phishing attacks according to rules (either static rules or fuzzy rules) either depending on the features of the URL or its content.

Abdelhamid et al. [10] use Associative Classification by defining simple ''If-Then'' rules to achieve a good accuracy rate for the classification of web pages. The rules are not static in the paper; they tested 6 different algorithms in their work. Each algorithm produces a different number of rules. However, it depends on third-party services to detect the age of the domain, web site traffic, DNS records etc.

The authors proposed a rule-based web page classification system by using only 10 rules for detecting zero-day phishing attacks [11]. The system uses a Support Vector Machine algorithm for classification, and it is independent of third-party services and languages. They focused on only Internet Banking pages, using 549 legitimate sites and 1158 unique e-banking websites, and implemented the detection system as a Chrome browser extension.

In [12], the authors implemented a rule-based incremental method that is lightweight and proactive in its detection of previously not encountered phishing URLs. Because of its computational intelligence and independence from blacklist signatures, this tool is extremely accurate when it comes to detecting zero-day and spear phishing exploits.

### B. VISUAL SIMILARITY BASED DETECTION SYSTEMS

The main deceptive aspect of a phishing webpage stems from its high visual resemblance to the legitimate page in terms of texts, fonts (by style, size, and color), used images (by their features such as height and width, and their positions), page layouts, HTML tags, etc. [13]. Some phishing detection systems use visual similarity-based techniques by utilizing these resemblance features to make the classification as reviewed in [13]. After making a feature-based comparison, if the similarity is greater than the predefined threshold value, it is identified as a fake/phishing webpage.

The authors of [14] visually compared a suspicious web page to a legitimate one to see if they were more similar than an acceptable rate. To check web pages, they used three-page features such as text features, image features, and the overall visual appearance of the page.

In [15], to detect the phishing pages, authors used visual characteristics of the web page by measuring the suspicious pages' similarity to legitimate ones. They checked the block level similarity, layout similarity, and style similarity for this detection process. If the system calculates a similarity value that is greater than the threshold value, it generates an alert report for the user. The system was tested with only a 328-page dataset. According to this dataset, they achieved satisfactory results.

As demonstrated by the preceding research, there is no globally accepted dataset for making meaningful comparisons. In many projects, authors generate their own dataset with a limited number of data points. An extended dataset is used in relatively new work [16], with 1,367 legitimate pages and 2,129 real phishing pages. In this work, they used global and local features of the webpage. Apart from that, this work relies on image level comparison by using a snapshot image of the legitimate webpage and the suspicious webpage. As a result, it can effectively deal with tricks that get help from the use of flash objects, images, or form of HTML content. Depending on the dataset, they reached a true positive rate of over 90 % and a true negative rate of over 97 %.

The primary advantage of these approaches is that they can quickly detect embedded objects such as images, Java Applets, ActiveX and Flash, which are located in HTML text. Additionally, they generally use signatures to identify the fake pages by taking into account the common features of the whole website.

### C. THIRD PARTY BASED DETECTION SYSTEMS

Due to its simplicity and ability to reach an acceptable accuracy rate, some authors developed their system by getting help from third-party services such as search engines/rankings or searching the whois/DNS records of the web page [17], [18], [19]. With these systems, they can check the legitimacy of the webpage and the authenticity of IP addresses and their associated domain names.

However, the false positive rate of these methods is relatively high because if a legitimate web page is newly constructed, it does not appear in the search results or returned values, which have a negative effect on this detection process. Additionally, a legitimate domain name can host phishing web pages on its host, which is hard to catch for this type of system.

### D. MACHINE LEARNING BASED DETECTION SYSTEMS

In recent years, there has been an increasing trend in the use of machine learning (ML) techniques, especially for solving lots of real-world problems. These techniques train the system with the help of a classification algorithm that uses a few features, like the page content, URL, and/or network features, to tell the difference between a legitimate website and a phishing website.

In [20], Karim et al. proposed a hybrid model to improve the accuracies of the machine learning models. The combination of linear regression (LR), support vector classifier (SVC) and decision tree (DT) is created using soft and hard voting methods and named as (LR + SVC + DT).

Prabakaran et al. proposed an enhanced Deep Learning-based phishing detection approach that combines the strengths of Variational Autoencoders (VAE) and Deep Neural Networks (DNN) to effectively identify malicious URLs [21]. In [22], Alshingiti et al. proposed three different techniques including LSTM, CNN and an LSTM–CNN-based approach to identify phishing websites. They got the accuracies of the suggested techniques, i.e., 99.2%, 97.6%, and 96.8% for CNN, LSTM–CNN, and LSTM, respectively.

In the study [23], the authors proposed a URL-based phishing detection system by analyzing the login URLs. As a discriminative property, they focused on homepages with login forms by emphasizing that the current systems have a high positive rate when tested with the legitimate login pages. They detected a total of six distinct phishing web domains, and their findings were contingent on the kind of service used by the attacker. Automatic feature extraction based on ''Term Frequency Inverse Document Frequency'' at the character N-gram level paired with the Logistic Regression algorithm was one of their methods of choice in the implementation and they achieved valuable results.

The used datasets are generally imbalanced due to the large number of legitimate web pages. Therefore, He et al. [24] focused on the detection of phishing attacks by using the cost sensitive XGBoost method. To balance the dataset, they preferred the use of the synthetic minority over-sampling technique (SMOTE) for oversampling. By using a crawler deployed in their lab, they randomly selected 600,000 URLs, 400,000 benign and 200,000 malicious in their dataset. However, there are no details about the dataset, and they are also not shared publicly.

Some of the authors concentrated on IoT device security by blocking phishing URLs [25]. They used an autoencoder mechanism to increase the performance of the proposed systems. They also tested different optimizers and said that the Adam optimizer gave the best results for this application area.

Some studies have tried to obtain more successful models by combining different deep learning architectures. In [26], they proposed a customized model consisting of CNN and LSTM layers. 3 different feature sets are used in this study. These are lexical features, character-level emphases, and word embeddings. Different models were built on these feature sets, and the outputs of these models were combined into one model. Lexical features are obtained because of feature engineering. 30 different hand-crafted lexical features are used. A balanced dataset was used in the study. The total size of the dataset is approximately 2.5 million.

A phishing URL detection system based on the RNN architecture was developed in [27] and tests were done using only URL information. There are no manually created features. The feature that distinguishes this study from others is its ability to visualize the effects of the URL sections on the outcome. Thanks to the visualization they've developed, which fields in the URL cause phishing characters can be understood by the human eye. LSTM, GRU, BiLSTM, and BiGRU were tested in the study and achieved 0.99 accuracy on the 1.5M dataset.

There is no widely used dataset accepted in the literature yet. Some of the researchers collect their datasets from open-source sources, and some of them use the data collected in earlier studies. In [28], the authors created a dataset that they collected and performed training and test operations on this dataset. In addition, they made comparisons using 3 different public datasets in previously published studies in literature. In the study, which was made with the CNN architecture, both traditional machine learning methods and many deep learning methods were tested and compared.

They proposed an approach in [29] that combines CNN and attention based hierarchical RNN architecture. An unbalanced dataset was collected and used within the scope of the study. 4.5M legitimate samples and 241K phishing samples were used. In this study, an accuracy value of 0.97 was obtained.

In [30], a phishing URL detection system was developed using machine learning techniques. In this study, hand-crafted was used. A dataset of 73,575 URLs with a balanced class distribution is used with 0.978 accuracy. A CNN-based approach is suggested in the study [31]. The study developed in [32] can be used as a real-time system and can be improved according to user feedback. A system working in real-time has been developed with a Chrome extension. In this study, besides traditional machine learning approaches, RNN-based GRU and LSTM approaches were tested. The accuracy rate

of this study was 0.99. The dataset used in this study was collected from open sources. The authors aimed to utilize features for developing a phishing detection system and designing an automated, complete, real-time system. A system working in real-time has been developed with a Chrome extension. In this study, besides traditional machine learning approaches, RNN-based GRU and LSTM approaches were tested. The accuracy rate of this study was 0.99. The dataset used in this study was collected from open sources. On the other hand, [33] aims to utilize features for developing a phishing detection system and designing an automated, complete, real-time system. Pre-processing, clustering, feature selection, classification, and k-fold cross validation were applied in the study. This study uses web page information, not only URL information, to enhance the accuracy rate. A dataset consisting of 500 web page records in total was used in the study and 0.963 accuracy was achieved. Reference [34] proposes a data-driven approach to detect phishing websites using various machine learning classifiers, such as Decision Tree, XGBoost, Random Forest, Support Vector Machine, and Naive Bayes by implementing various numbers/types of features such as URL-based features, hyperlink-based features, and hybrid features. They developed a dataset with 6000 URLs containing 3000 legitimate URLs and 3000 phishing URLs. They collect web page information with a crawler and apply feature extraction to the data collected. Experimental results on the dataset show that the XGBoost algorithm with hybrid features has a 98.81% true positive rate and 0.49% false-positive rate.

The spread of phishing attacks over email is quite common. It is also possible to detect phishing via email content. In [29] and [30] the authors developed a study that detects phishing attacks using email content. It is a confusing system to adapt to the real world because e-mail is mostly considered confidential. The study developed in the former one achieved 0.98 accuracy with the dataset consisting of 10K samples. In [36], only traditional machine learning algorithms were evaluated while [35] focused on machine learning methods and deep learning-based CNN and RNN architectures. Preprocessing, clustering, feature selection, classification, and k-fold cross validation were applied in the study. This study uses web page information, not only URL information, to enhance the accuracy rate. A dataset consisting of 500 web page records in total was used in the study and 0.963 accuracy was achieved. The detailed comparison of the literature on machine learning/deep learning-based phishing detection systems is shown in Table 1. In this research, it is aimed to use deep learning-based solutions with the dataset that is known as the greatest up to now.

## III. BACKGROUND

The proposed framework processes the URL address of the web page and classifies them either as phishing or legitimate by using a deep learning-based approach.

A background knowledge is required to understand the rest of the article clearly. These are explained in this section.

### A. URL ADDRESSES

The address of a web page is named as Uniform Resource Locator (URL), which uses mainly the HTTP or HTTPS protocol. A URL is essential for specifying the location of any type of resource, such as an HTML page, file, image, directory, program, etc. The URL components, as depicted in Fig. 4, play a critical role in understanding the used methodology.


**FIGURE 4.** URL components.

A URL may contain some or all of the following components.

- A hostname consists of three parts: A domain name is always present in the URL, and it is used for locating the computer on the Internet. It is a registered identification ''string'' and is used instead of numeric IP addresses.
- A top-level domain is also always present in the URL. It shows the shortened form of an organization type as ''.com'', ''.org'', ''.net'' etc. in the hierarchical structure of the Domain Name System (DNS). A subdomain is a part of a larger domain that has a specific interest area and services.
- The protocol is always present, but not always visible in the URL and it determines how data is transferred between the server and client side, mainly HTTP and HTTPS; additionally, SMTP, FTP, DHCP, etc.
- Path/File shows where the page or file is on the website. The notation of the path/file depends on how the website is set up.
- HTML anchors or fragments are used on the webpage to show the internal page navigation.

Parameters are located at the end of the URL; they are represented with key/value pairs, which use '?' and '&' characters respectively.

### B. DEEP LEARNING

While the capability of computers has increased, Artificial Intelligence (AI) has become an expanding concern in computer science due to its aim of making computers intelligent like humans. Learning is perhaps the distinguishing factor that makes us human.

Therefore, researchers focused on how to program this factor for computers, and a new subfield of AI emerged as machine learning (ML). An ML algorithm enables forecasting future things without rules and models, building models to explain the execution styles of real-world entities, and identifying patterns according to observed data.

Deep learning is one of the algorithms or subsets of machine learning. It primarily refers to deep artificial neural networks or deep reinforcement learning, both of which use

**TABLE 1.** Comparison of the literature.

| Year/Ref | Dataset | Applied Models | Metrics (Acc.) | Pros | Cons |
|---|---|---|---|---|---|
| 2023/20 | 11,054 URLs with 32 features | Hybrid LSD model (LR+SVC+DT | 0.981 | Login URLs are used. | No Deep Learning |
| 2023/21 | 79,745 training<br>19,913 testing | Variational Autoencoders (VAE) and Deep Neural Networks (DNN) | 0.974 | Not depend on external services | Long response time |
| 2023/22 | 20,000 URLs, 80% for training<br>20% for testing | LSTM, CNN, and LSTM–CNN | 0.992 | High accuracy value | Does not check the status of the URL if it is active or not, small dataset |
| 2022/23 | 60,000 legitimate<br>30,000 malicious | LightGBM, XGBoost, AdaBoost, RF, SVM, kNN, NB, LR. | 0.965 | Login URLs are used.<br>Not depend on external services | High similarity between legitimate and phishing pages<br>No Deep Learning |
| 2021/24 | 200,000 malicious<br>400,000 legitimate | XGBoost, Cost Sensitive XGBoost,<br>SMOTE XGBoost | 0.99 | Focused on an imbalanced dataset | Dataset is not public, Limited number of Data, No Deep Learning |
| 2022/25 | 35000-benign<br>12000-spam<br>10000-phishing<br>11500-malware<br>45000-defacement | DNN combined with Autoencoder | 0.989 | Aims for the detection of attacks on IoTs.<br>Uses a multi-classification dataset | Data collected from different resources with different classes,<br>Needs more real-time data |
| 2020/26 | 1.3M malicious<br>1.2M legitimate | Custom model combined multiple CNN and LSTM layer | 0.944 | Higher accuracy on the same dataset than the referenced study. | Single model Neural Network architecture used |
| 2020/31 | 36,400 legitimate<br>37,175 Phishing | CNN | 0.98 | Higher accuracy rate compared to the reference study | No different model and results for comparison. |
| 2020/27 | 800,000 legitimate<br>759,485 Phishing | RNN (LSTM, GRU, BiLSTM, BiGRU) | 0.99 | URL analysis can be done with the visualization technique | Only RNN-based architectures have been tested. Dataset is not public. |
| 2020/28 | 4 different datasets<br>279435 Legitimate<br>279527 Phishing | Naïve Bayes, LR, RF, XGBoost, RNN, RCNN, DNN, VDCNN, CNN | 0.985 | Experiment on 3 public datasets and their own datasets | Training time is long |
| 2019/29 | 241,047 phishing<br>4,579,893 legitimate | RNN, CNN, RNN+CNN | 0.979 | Novel Objective function | Quite little Phishing data |
| 2019/30 | 36,400 legitimate<br>37,175 phishing | RF, NB, kNN, kStar, Adaboost, Decision Tree, SMO | 0.978 | Detection using only URL information | The need for feature engineering |
| 2022/32 | 429,125 legitimate<br>236,362 phishing | Logistic Regression, SVM, RF, RNN-GRU, RNN-LSTM | 0.991 | with chrome extension in real-time environment without delays () | only RNN-based architectures have been tested. |
| 2021/33 | 500 records in total | Modified crow search-based deep learning NN, ANN, SVM, KNN, ANFIS | 0.963 | Uses web page content not only URL information | Small dataset for training and evaluation |
| 2022/34 | 3,000 Phishing<br>3,000 legitimate | Decision Tree, XGBoost, Random Forest, SVM, and NB | 0.991 | Use web page content in feature extraction not only URL information | Hand-crafted features on relatively small dataset |
| 2022/35 | 3004 phishing emails<br>7234 legitimate emails | DNN, CNN, LSTM, Random Forest, DLF: PUD, MFP, GA | 0.981 | Focused on email data | Hard to adapt to the real-time environment. Because e-mail is mostly considered confidential. |
| 2022/36 | 8,351 phishing<br>8,400 legitimate<br>2500 ham<br>500 spams | RF, NN, Averaged perceptron, LR, Boosted decision tree, SVM, locally deep SVM | 0.977 | Capable of detecting phishing emails | It is a confusing system to adapt to the real-time environment. Because e-mail is mostly considered confidential. |

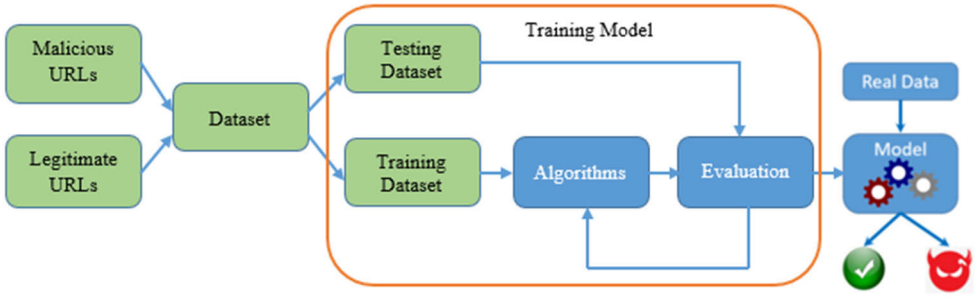


**FIGURE 5.** The flowchart of DEPHIDES.

the term ''deep'' to refer to the number of layers in the system [37]. There are different methods (such as convolutional neural networks, recurrent neural networks, long short-term memory, and auto-encoders) to apply DL, and these methods depend on the use cases, such as the kind of data that needs to be processed.

One of the main differences and applicability of DL and ML is their data dependencies. ML algorithms are preferable and result in well even if there is a small dataset. However, DL is a ''Data Hungry'' approach, and if you have more data, you will get more accurate results. Due to its structure and depth, DL needs more computation. However, with the emergence of new computation technologies such as the graphical processing unit (GPU), parallel computation is possible with the use of thousands of cores compared to a single CPU, which has a small number of cores.

Although DL methods generally use the power of parallel computation, the extended use of big data results in increased training time, which ranges from hours to weeks to months. However, traditional ML algorithms often train fast, which ranges from a few minutes to hours.

## IV. DEEP LEARNING BASED PHISHING DETECTION SYSTEM (DEPHIDES) DETAILS

In the proposed model we made a data mining process according to the flowchart in Fig. 5. Initially we collected a relatively big dataset from Internet sources.

The dataset is divided into three parts for making a comparable performance measurement. 70 % is used for training, while 20 % is for validation and 10 % is used for testing.

By using this flowchart, the system is trained for five different deep learning algorithms as Artificial Neural Network, Convolutional Neural Networks, Recurrent Neural Networks, Bidirectional Recurrent Neural Networks, Attention Networks. Finally, the trained model can be used in real world implementation for detection of phishing web pages.

### A. DATASET COLLECTION

Within the scope of this study, a comprehensive dataset has been created and several experiments have been run on this dataset. Phishing URLs are collected from ''www.phishtank.com'', and legitimate URLs are collected from ''commoncrawl.org'', in the dataset created. PhishTank is an open-source community where phishing URLs are shared and kept up to date. Shared URLs are subject to a vote. A URL submitted because of voting is assigned to one of three classes. These classes are Valid Phish, Invalid Phish, and Unknown. A URL is marked as valid phishing if enough votes are collected to show that it is phishing. Otherwise, it is assigned to the invalid phishing class if enough votes are collected showing that the URL is not phishing. URLs that cannot be identified as valid or invalid are marked as unknown. This control mechanism ensures high accuracy of collected phishing URLs. URLs in the Valid Phish class, which were shared by PhishTank until August 2018, were collected. In this way, 2,320,893 phishing URLs are included in the dataset created in this article. An example of the dataset is shown in Table 2.

To develop a machine learning application for the detection of phishing attacks, it is also necessary to collect URLs that are not phishing. We got help from the Common Crawl corpus which contains petabytes of data collected over the last 7 years. It contains raw web page data, extracted metadata, and text extractions. The concept that well-known legitimate domains receiving a considerable number of backlinks tend to have high rankings is a valid observation. It's challenging for a domain to rank prominently in search engine results if it is being used for malicious purposes, as search engines aim to provide users with the most relevant and trustworthy content. Therefore, the inclusion of such domains in the legitimate URL dataset ensures the reliability of the data collected.

To compile the legitimate URL dataset, the first 100,000 domains from PageRank rankings worldwide, calculated by Common Crawl using data from web crawling, were utilized. For each of these domains, approximately 30 random URLs were selected, resulting in a collection of 2,881,948 legitimate URLs. This extensive dataset provides a robust foundation for the development and testing of phishing detection systems, as it represents a wide array of legitimate online resources.

The developed phishing detection system adopts an approach that involves collecting URLs from domains that are unquestionably legal. Typically, the URL structure within a

**TABLE 2. Data samples.**

| Type | URL |
|---|---|
| phishing | http://direct-certs.bankofamerica.com.techdbaseurl46.cn/direct/certpickup.asp?session=5205817050667567602066511025245896499574920702005427213141662attackers |
| legitimate | https://www.aai.aero/en/airports/passenger-info/varanasi/eat-&-dine |

domain remains consistent and standard. However, to ensure that the algorithm does not become overly specialized or memorize the specific structure of a particular domain, only 10 or fewer URLs from such domains have been added to the dataset. This approach helps maintain a broader and more generalizable training dataset, allowing the system to effectively detect phishing attempts across various legitimate domains and their respective URL structures.

**TABLE 3. Dataset details.**

| Dataset | Class | Small Dataset | Big Dataset |
|---|---|---|---|
| Train | Phish | 162,463 | 1,624,623 |
| | Legitimate | 201,736 | 2,017,363 |
| | Overall | 364,199 | 3,641,986 |
| Validation | Phish | 46,649 | 466,504 |
| | Legitimate | 57,927 | 579,271 |
| | Overall | 104,576 | 1,045,774 |
| Test | Phish | 22,978 | 229,766 |
| | Legitimate | 28,532 | 285,314 |
| | Overall | 51,510 | 515,080 |
| All | Phish | 232,090 | 2,320,893 |
| | Legitimate | 288,195 | 2,881,948 |
| | Overall | 520,285 | 5,202,841 |

In our study, approximately 5.1 million URLs were collected, with 2.3 million being phishing and 2.8 million being legitimate. As far as we know, the dataset is the largest shared dataset in literature for this purpose. To ensure the detection of the latest attacks and address the limitations of supervised learning, the iterative updating of the model with newly acquired training data proves highly beneficial. In a commitment to transparency and collaboration, all source code and datasets generated within this project have been shared as open source, which facilitates continuous improvement, allowing the model to be regularly updated with additions to the dataset, thereby enhancing its efficacy and adaptability over time.

There is a need for high hardware resources to process these amounts of data by machine learning algorithms. For this reason, two different versions of the dataset have been created for the tests performed in this article. The first of these versions (small_dataset) consists of a 10% subsample of the total dataset. This means 232,090 phishing, 288,195 legitimate URLs. The other version (big_dataset) includes the whole dataset as given in Table 3. Due to hardware constraints, most of the tests in this article were run on a small_dataset to compare the results obtained by other researchers doing research in this area.

### B. VECTORIZATION

Attackers employ various tactics in phishing attacks, some of which pose significant challenges for detection. The attackers employ techniques that pose significant challenges for conducting word-based analyses. This is due to the widespread use of subtle alterations that are hard for individuals to discern. To perform word-based analysis, one must undergo training on a corpus forming a substantial volume of nonsensical words. Furthermore, word-based analysis introduces a level of language dependence. Consequently, in this article, we employed a character-based embedding approach for the vectorization process. Since the embedding size for each char is determined as 50, the vectorized version of the dataset is as shown in Fig. 6.

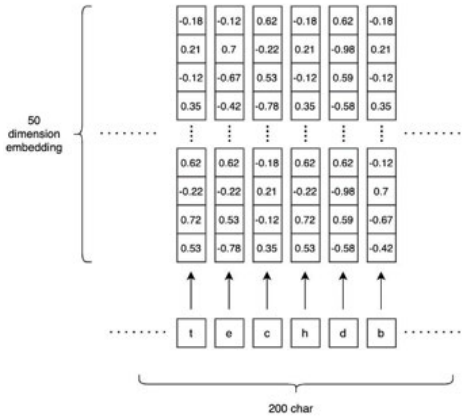


**FIGURE 6. An example of dataset vectorization.**

During the vectorization phase, it is essential to constrain the size of the URL to a specified character limit. To figure out the appropriate character limit, statistical metrics are evaluated, taking into account the character lengths of the URLs. Consequently, for each class, we calculate the minimum, maximum, standard deviation, and median separately. These statistical findings are presented in Table 4, with the data being derived from the extensive dataset.

It becomes clear that a URL length of 200 characters adequately encapsulates the majority of the dataset with

**TABLE 4.** Character length statistics for dataset.

| Class | Count | Average | Min | Max | Std. Dev. | Median |
|---|---|---|---|---|---|---|
| Phishing | 2,321,934 | 86.9 | 12 | 6,104 | 64.9 | 68 |
| Legitimate | 2,881,948 | 72.0 | 12 | 5,984 | 45.3 | 63 |
| All | 5,203,882 | 78.6 | 12 | 6104 | 55.4 | 65 |

more than 95% of the dataset. Consequently, this specific value is adopted in the vectorization process. URLs exceeding 200 characters are truncated to this length, while URLs shorter than 200 characters are padded to reach 200-character length. In Fig. 7, a histogram chart is presented, depicting the URL length distribution in the "big_dataset." The histogram is generated by grouping URL lengths into 20-character buckets.

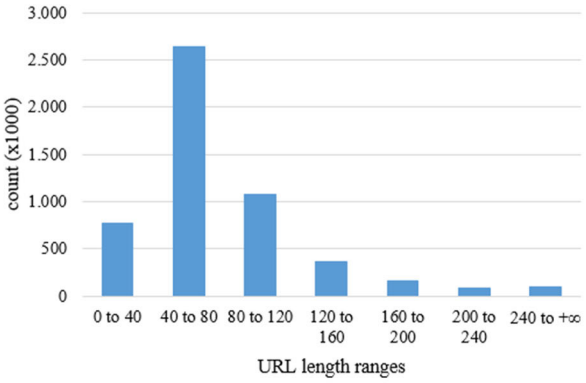


**FIGURE 7.** Histogram of the URL length ranges.

In our dataset, a consistent dimension is imposed during the training process, irrespective of the original sizes of URLs. URLs shorter than the specified length are padded to reach the designated dimension, while those exceeding the specified length are truncated to fit. This URL length parameter is applied uniformly to all records in the dataset. Selecting the best URL length entails striking a balance between the model's architectural complexity and its coverage. If the URL length is excessively long, it substantially prolongs both training and testing times, thus diminishing the model's practicality and efficiency. Conversely, an overly short URL length compromises the model's training performance. As a result, for this study, a URL length of 200 was chosen, as it is deemed well-suited for effectively representing the majority of the dataset.

Attackers may attempt to bypass prevention systems by utilizing URL shorteners. Even if the DEPHIDES URL checking service fails to successfully classify the shortened URLs, the attackers' objectives cannot still be reached because the shortened URLs are directed to original web pages during the attack, and they will be accurately classified as phishing by the system.

### C. OVERVIEW OF TESTED ALGORITHMS

This article presents the development of a phishing URL detection system based on deep learning. The dataset employed for training the model consists of sequential data, therefore the proposed model necessities the use of an algorithm, which is capable of processing such sequences. In addressing the phishing URL detection challenge, the approach involves treating the URL as a single entity and learning the relationships among its smaller sub-parts.This approach is widely used to make sense of textual data. For example, studies that perform semantic analysis have similarly been able to achieve good success by evaluating the sub-parts of the text together. In addition, it was thought that it would be effective to use the most commonly preferred algorithms for similar problems in literature. Due to the high complexity of certain algorithms, training them becomes a challenging task. Recognizing the critical importance of faster response in detecting cyber-attacks, there is a deliberate effort to avoid favoring algorithms with high complexity. A total of five distinct deep learning architectures were examined. Furthermore, analyses were conducted to enhance the accuracy rates by exploring various configurations for each architecture.

**Artificial Neural Network (ANN)** is characterized by its acyclic connections between units. In this approach, the architecture primarily includes dense layers. Each node performs a direct linear calculation, and the parameters are then updated based on the output of the loss function.

**Convolutional Neural Networks (CNNs)** belong to the realm of deep learning algorithms and are tailored to demand minimal preprocessing. They employ a form of multilayer perceptron and execute convolution operations by envisioning them as sliding window functions applied to a matrix. When it comes to word-based approaches, this window traverses through words in the input data. Each convolutional operation yields a response when it detects specific patterns. These patterns may encompass expressions such as "I hate" or "very good," enabling CNNs to identify them within a sentence irrespective of their position. Character-based approaches are instrumental in recognizing malicious patterns in URLs used in phishing attacks. In Section III-B, we delved into the challenges associated with identifying phishing attacks.

**Recurrent Neural Network (RNN)** constitutes another subset of deep learning techniques, where connections between nodes create a directed graph along a sequence. RNN can be envisioned as a sequence of interconnected neural network blocks, forming a chain-like structure. It can be likened to multiple replicas of the same network, each passing information to its successor. This architecture is particularly well-suited for handling sequential data, such as text.

In the RNN framework, when processing a word, it uses the knowledge of preceding words. Each word takes the output

**TABLE 5.** Summary of architectural components of each algorithm used in scenarios.

| Algorithm | Architecture | Source Code |
|---|---|---|
| **Base Scenario** | | |
| ANN | Embedding > Dense > Flatten > Dense | [38] |
| ATT | Embedding > SeqSelfAttention > Flatten > Dense | |
| RNN | Embedding > LSTM > Dense | |
| BRNN | Embedding > BidirectionalRNN > Dense | |
| CNN | Embedding > Conv1D > Flatten > Dense | |
| **Complex Scenario** | | |
| ANN | Embedding > Dense > Dense > Dense > Dense > Dense > Dense > Dense > Flatten > Dense | [38] |
| RNN | Embedding > LSTM > LSTM > LSTM > LSTM > LSTM > LSTM > LSTM > Dense | |
| CNN | Embedding > Conv1D > MaxPooling1 > Dropout > Conv1D > Dropout > Conv1D > Dropout > Conv1D > MaxPooling1 > Dropout > Conv1D > Dropout > Conv1D > MaxPooling1 > Dropout > Conv1D > MaxPooling1 > Dropout > Flatten > Dense | |
| BRNN, ATT | cannot be performed due to the time complexity | - |
| **Complex CNN Scenario** | | |
| CNN1 | Embedding > Conv1D > MaxPooling1 > Dropout > Conv1D > Dropout > Conv1D > Dropout > Conv1D > MaxPooling1 > Dropout > Conv1D > Dropout > Conv1D > MaxPooling1 > Dropout > Conv1D > MaxPooling1 > Dropout > Flatten > Dense | [38] |
| CNN2 | Embedding > Conv1D > MaxPooling1 > Dropout > Conv1D > Dropout > Conv1D > Dropout > Conv1D > Dropout > Conv1D > Dropout > Conv1D > Dropout > Conv1D > MaxPooling1 > Dropout > Conv1D > Dropout > Conv1D > Dropout > Conv1D > Dropout > Conv1D > MaxPooling1 > Dropout > Flatten > Dense | |
| CNN3 | Embedding > Conv1D > MaxPooling1 > Conv1D > Conv1D > Conv1D > MaxPooling1 > Conv1D > Conv1D > Conv1D > Conv1D > Conv1D > Conv1D > Conv1D > MaxPooling1 > Conv1D > Conv1D > MaxPooling1 > Conv1D > Conv1D > Conv1D > Flatten > Dense | |
| **Big Dataset 100 Epoch** | | |
| CNN1 | Embedding > Conv1D > MaxPooling1 > Dropout > Conv1D > Dropout > Conv1D > Dropout > Conv1D > MaxPooling1 > Dropout > Conv1D > Dropout > Conv1D > MaxPooling1 > Dropout > Conv1D > MaxPooling1 > Dropout > Flatten > Dense | [38] |

of the previous words as its input. RNN allows for one-way information flow, while Bidirectional Recurrent Neural Networks (BRNN) facilitate two-way information exchange. This means that not only is information from preceding words used in processing a word, but the subsequent words are also taken into account during the process.

**Bidirectional Recurrent Neural Network (BRNN)** architecture operates on the principle of splitting the neurons found in a regular RNN into two directions: one for the positive direction and another for the negative direction. Traditional RNNs have unidirectional information flow, meaning they only consider information from preceding words when processing a word. However, for comprehensive word interpretation, it's often necessary to incorporate both preceding and subsequent words. BRNN accomplishes this by running one RNN in a forward direction from the beginning and another in reverse from the end.

In contrast, **Attention Networks (ATTs)** employ an approach that prioritizes comprehending the entire context rather than focusing on isolated components. "Attention" in this context refers to the conscious directing of the mind toward a specific object or aspect. Attention networks discern critical portions of the data and incorporate this information into their memory, which is subsequently utilized in subsequent steps. These networks pinpoint and select the important or distinctive elements of the data. This means that neural networks make informed decisions regarding which features deserve their attention, thus assigning more weight to essential elements across the entire dataset.

Many deep learning approaches eliminate the need for the feature engineering step. Basic keyword searches are a method that can be easily avoided. For instance, instead of using amazonsecurelogin.com, a similar domain like amazonserucelogn.com could be employed for attacks. Since the words "secure" and "login" are not used directly in this example, keyword checking may not be successful. Furthermore, attackers can not only target English but also launch regional attacks by creating URLs based on the language of that region. This approach will impose additional tasks on keyword checking systems, such as gathering, using, and updating an extensive word cloud covering multiple languages. Having a specific phishing keyword may not always provide useful information for classifying the URL as phishing. For instance, a URL like amazon.com/customerservices/login may legitimately serve a legal purpose. Although deep learning algorithms have shown success in making such inferences.

URL hijacking, also known as URL redirection or URL spoofing, is a type of cyber-attack where an attacker redirects a website's traffic from a legitimate URL to a malicious one. However, the proposed system cannot detect URL hijacking attacks due to its structure.

It is not necessary to retrain the entire dataset when incorporating new data. The model can be updated iteratively, continuing from where it left off in the previous training.

In this article, in addition to deep learning-based approaches, tests were conducted with traditional machine learning algorithms and the results were compared.

- Random Forest (Ensemble Based Algorithm)
- Naïve Bayes (Statistics Based Algorithm)
- Logistic Regression (Statistics Based Algorithm

Architectural components were used in the Keras tool. A custom architectural design was made for each scenario using architectural components. Table 5 provides summaries of the designed architectures. The detailed architectures of all scenarios can be accessed by clicking on the link given in each row at the table.

## V. EXPERIMENTAL RESULTS

In the context of this study, a deep learning-based system was developed for the purpose of detecting URLs used in phishing attacks. The effectiveness of this system was assessed by comparing its performance with traditional machine learning algorithms through a series of tests. In the implementation of deep learning architectures, the Keras Framework was used in the front end, with the backend powered by the Tensorflow Framework. Traditional machine learning algorithms were implemented and tested using the Scikit-Learn package, which was also used for reporting test results.

The evaluation of the system's performance entailed the use of several key metrics, including accuracy, loss, precision, recall, and F-score. A detailed explanation of these metrics can be found in Section III.E. In addition to these metrics, the study also considered the execution time of the tests. Performance evaluation is a crucial aspect in comparing the accuracy of different algorithms, and execution times may vary based on the underlying hardware. For this project, a server from Floydhub was leased for conducting the tests. The leased server is equipped with a Nvidia Tesla V100 model GPU. V100 has 5120 cores and 16 GB of VRAM. The operating frequency of each core is 1370 Mhz.

Although the focus of this study was to develop a deep learning-based system, the results were compared with those of traditional machine learning algorithms. Hyperparameters should be determined for the use of deep learning algorithms. Common hyperparameters used in the tests carried out within the scope of this study are:

- Loss function: Binary Cross Entropy
- Sequence size: 200
- Update function and its parameters: ADAM; Learning rate: 0.01, Beta_1: 0.9, Beta_2: 0.999.
- Character embedding dimension: 50

At the stage of creating architecture, parameters such as the number of layers and number of neurons are given. The default values of the frameworks used in other hyperparameters are used.

This article investigated the performance of five distinct deep learning architectures: ANNs, CNNs, RNNs, BRNNs, and ATTs. These architectures exhibit varying memory and processing power requirements. Some may demand less memory while others consume more, and the same principle applies to processing power required during training.

To ensure optimal GPU utilization and an equal accelerator effect across all architectures, the batch size parameter was fine-tuned in the tests. In cases where processing the entire dataset at once posed a challenge, the dataset was divided into smaller portions, each of which was processed sequentially. It's important to note that larger batch sizes consume more power and memory. In the tests conducted as part of this study, adjustments were made to the batch size parameter to maximize GPU utilization for all test scenarios. Importantly, altering batch size values did not have a discernible impact on the accuracy and loss values. Instead, this parameter primarily affects the volume of data processed within a given timeframe, without influencing the overall quality of the results.

### A. DEEP LEARNING APPROACHED WITH BASE SCENARIO

To facilitate a comprehensive comparison of running times and architectural performance, all five architectures underwent testing with a common configuration featuring a single layer of 128 neurons. This configuration served as the baseline, and subsequent performance-enhancing techniques were applied to enhance overall success. The tests for this configuration spanned 20 epochs, and the results are visually depicted in Fig. 8, and batch size, train accuracy, validation accuracy and test accuracy values are shown in Table 6.

**TABLE 6. Performance metrics for base scenario with 20 epochs.**

| Arch. | Batch Size | Train Acc. | Val. Acc. | Test Acc. | Avg. Train Time (sec) |
|---|---|---|---|---|---|
| RNN | 7000 | **0.950** | **0.952** | **0.951** | 38.36 |
| BRNN | 7000 | 0.949 | 0.947 | 0.947 | 150.98 |
| CNN | 7000 | 0.914 | 0.915 | 0.914 | 4.85 |
| ANN | 7000 | 0.915 | 0.916 | 0.916 | **2.34** |
| ATT | 1000 | 0.901 | 0.901 | 0.900 | 124.50 |

The findings presented in Table 6 revealed that the RNN architecture emerged as the most successful deep learning architecture, boasting an impressive test accuracy of 95.1%. It's worth noting that accuracy rates among the architectures were relatively close, but RNN led the pack. Interestingly, both RNN and BRNN architectures exhibited lower initial validation accuracy rates compared to the other architectures.

Beyond accuracy rates, the evaluation of algorithm performance also considered the runtime durations. In the baseline

scenario, the tested architectures underwent training over 20 epochs, and the average durations for each epoch can be found in Table 6.

Based on the obtained results, it is evident that recurrent algorithms (RNN, BRNN, and ATT) operate at a notably slower pace compared to the other architectures. Among these, the BRNN architecture proved to be the slowest, with an average time of 150 seconds per epoch.

### B. DEEP LEARNING APPROACHED WITH COMPLEX SCENARIO

To enhance the likelihood of accuracy, one approach is to increase the complexity of deep learning architectures or extend the number of training epochs. In the subsequent phase, more intricate architectures were examined, and their outcomes were assessed. All architectures subjected to testing were designed with seven layers during this phase. Determining the number of neurons in the model architecture is a critical consideration. Augmenting the number of neurons introduces complexity to the model, resulting in a larger number of parameters that must be learned during training.

The complexity of a model with increased training and testing times is a crucial trade-off. Conversely, setting the number of neurons too low can impede the model's ability to learn effectively. When determining the number of neurons, careful consideration of these factors is essential to make an informed choice. Given that the number of neurons is a numerical parameter independently adjustable for each layer, a wide range of values is possible. Different architectures have their unique neuron structures, resulting in varying complexities between, for instance, RNN and CNN neurons. To ensure a controlled experiment across all deep learning architectures, a consistent number of neurons (128) was employed for all layers. This choice was made to ensure successful testing while maintaining reasonable training times for all architectures. After thorough research and preliminary testing, it was determined that using 128 neurons in all layers was a well-balanced decision. The experiments in this study were also trained over 20 epochs, and the test results for the architectures can be found in Fig. 9.

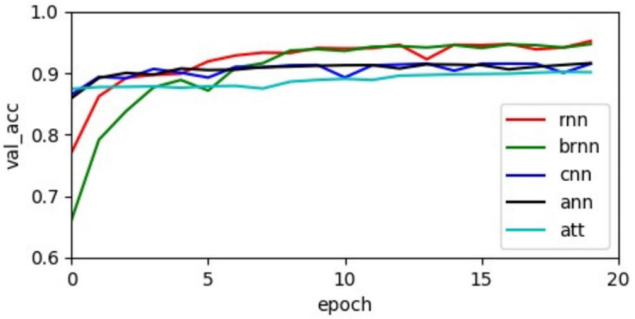


**FIGURE 8.** Base scenario's validation accuracies.

Apart from other performance metrics, running times play a significant role in architecture selection. While tests were executed on GPUs with better performance, architectures still required a substantial amount of time for testing. Due to hardware limitations, not all architectures could be assessed using complex configurations. In this stage of the study, tests were performed on three architectures. For example, RNN architecture took approximately four hours to complete 20 epochs of training on the GPU. Based on the runtimes presented in the base scenario (Table 6 ), the BRNN architecture was roughly four times slower than the RNN architecture. Meanwhile, the ATT architecture lagged the RNN architecture by approximately 3.2 times. In this study, it wasn't feasible to provide the BRNN and ATT architectures with sufficient resources for training over 20 epochs, given the imperative need for extended training durations as seen in Table 7.

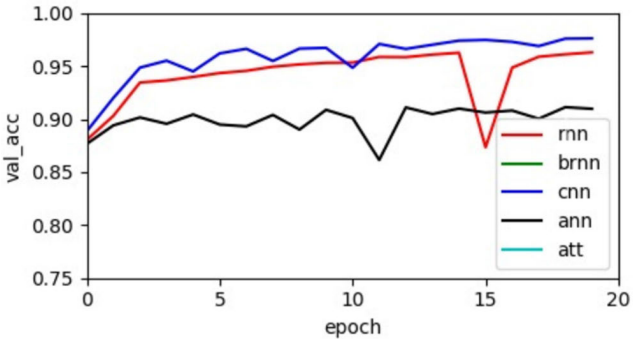


**FIGURE 9.** Complex scenario validation accuracies.

**TABLE 7.** Performance metrics for complex scenario with 20 epochs.

| Arch. | Batch Size | Train Acc. | Val. Acc. | Test Acc. | Avg. Train Time (sec) |
|---|---|---|---|---|---|
| RNN | 3000 | 0.962 | 0.962 | 0.963 | 709.39 |
| BRNN | Cannot be performed (due to need of huge GPU memory) | | | | |
| CNN | 7000 | **0.973** | **0.976** | **0.975** | **10.75** |
| ANN | 7000 | 0.911 | 0.909 | 0.910 | 11.48 |
| ATT | Cannot be performed (due to need of huge GPU memory) | | | | |

The results indicate that both the CNN and RNN architectures demonstrated superior performance on the test set compared to the ANN architecture. Furthermore, the CNN architecture outperformed the RNN architecture, achieving more successful results in both the test set and the validation set. Additionally, the CNN architecture exhibited a substantially shorter total running time, completing 20 epochs in about eight minutes, whereas the RNN architecture took approximately 238 minutes, encompassing vectorization and other preprocessing tasks. In the complex scenario tests, there was no substantial improvement observed in the ANN architecture, but accuracy values increased for the CNN and RNN architectures. These results suggest that the architectural complexity in CNN and RNN approaches contributes positively to accuracy metrics. Notably, the complex scenario

tests indicated increased running times for all tested architectures, highlighting the effect of architectural complexity on running time.

### C. COMPLEX SCENARIO DIFFERENT CNN ARCHITECTURES

The study concludes that as one of the Deep Learning Approach, CNN architecture is more suitable for phishing URL detection than the other architectures tested. As depicted in the graphs in Fig. 10, it is apparent that training has not yet reached a convergent state. Consequently, there is potential to achieve higher performance values with longer training sessions. Therefore, the tests conducted in the next phase encompassed 100 epochs. Additionally, two more CNN architectures were examined in this experiment, and their outcomes were compared. The performance metrics for the tests conducted on three complex CNN architectures with smaller datasets are presented in Fig. 10.

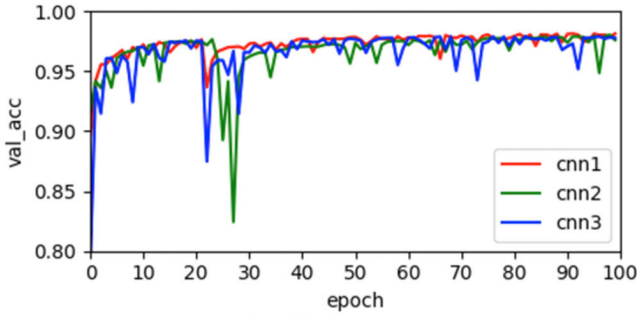


**FIGURE 10.** Complex CNN architectures' validation accuracies with 100 epochs.

In the testing process, the performance metrics are detailed as in Table 8, where the batch size was consistently set to 7,000 for all tests in this scenario. CNN1 is designed with lower complexity, featuring a 7-layer architecture.

**TABLE 8.** Performance Metrics for Complex CNN Architectures with 20 epochs.

| Arch. | Val. Acc. | Val Loss | Test Acc. | Test Loss | Avg. Train Time (sec) |
|---|---|---|---|---|---|
| **CNN1** | **0.981** | **0.051** | **0.980** | **0.051** | **10.9** |
| CNN2 | 0.976 | 0.064 | 0.976 | 0.063 | 22.0 |
| CNN3 | 0.977 | 0.086 | 0.977 | 0.085 | 17.6 |

The architecture that emerged as the most successful in both the validation and test sets was CNN1. Notably, this architecture also proved to be the fastest one when compared to the other architectures. In the complex scenario, the CNN architecture was identified as the most successful among the tests conducted over 20 epochs. The accuracy rates of the CNN architecture in the complex scenario were 97.6% in the validation set and 97.5% in the test set. It's important to note that the CNN architecture used in the complex scenario is the same as the CNN1 architecture employed in the complex CNN scenario. The distinction lies in the training duration: the complex CNN scenario underwent training for 100 epochs, whereas the other scenario was trained for 20 epochs. Upon examining the values in this table, it becomes evident that longer training periods had a positive impact on the CNN1 architecture. In the tests conducted with the complex CNN architecture over 100 epochs, the accuracy rate further increased to 98.0% in both the test and validation sets. Following the conducted tests, CNN1 was selected as the most successful architecture. This architecture was tested with 100 epochs of training on the big dataset, and the results were evaluated.

**TABLE 9.** Performance metrics of CNN1 architecture with 20 epochs.

| Class | Precision | Recall | F1-Score | #Sample |
|---|---|---|---|---|
| Phishing | 0.98 | 0.99 | 0.99 | 285,314 |
| Legitimate | 0.99 | 0.98 | 0.99 | 229,766 |
| **Average** | **0.99** | **0.99** | **0.99** | **515,080** |

Table 5 provides a breakdown of the architectures for CNN1, CNN2, and CNN3. The differences between CNN1, CNN2 and CNN3 are the factors that affect the architectural complexity, such as the number of layers and the number of neurons as depicted in Table 5. Among these three architectures, CNN1 distinguishes itself with its lower complexity, resulting in faster training times compared to the others. While increasing model complexity can enhance performance in deep learning-based problems, it is a judgment that doesn't hold universally true and may not apply in all cases.

Interestingly, among the tested complex architectures, CNN1, with its lower complexity, yielded the most successful results.In a test conducted on the big_dataset, both the validation and test accuracies achieved an impressive value of 98.74%. This indicates the system's robust performance and reliability in detecting phishing attacks. Moreover, during the tests conducted on complex CNN architectures with the smaller dataset, a test accuracy of 98.0% was obtained. However, when using the big_dataset, the test accuracy rose to an even higher 98.74%. This demonstrates the significant positive impact of utilizing a larger dataset on the system's overall performance.

Precision, recall, and F1-score are essential metrics for evaluating performance in classification problems. These metrics are provided in Table 9 and Table 10, differentiating between phishing and legitimate classes, offering a comprehensive view of the system's performance.

Accuracy values are changed depending on the iteration (epoch) numbers and it was shown in Fig. 11. These findings underscore the system's ability to distinguish between the two classes with a high degree of accuracy and minimal confusion.

For measuring the performance of the system GPU performance and its resource usage is also critical metric. The GPU

**TABLE 10.** Performance metrics of CNN algorithms with 20 epochs.

| Arch. | Precision | | Recall | | F1-Score | |
|---|---|---|---|---|---|---|
| | Phishing | Leg. | Phishing | Leg. | Phishing | Leg. |
| CNN1 | 0.98 | 0.98 | 0.98 | 0.98 | 0.98 | 0.98 |
| CNN2 | 0.97 | 0.99 | 0.99 | 0.96 | 0.98 | 0.97 |
| CNN3 | 0.98 | 0.97 | 0.98 | 0.98 | 0.98 | 0.98 |

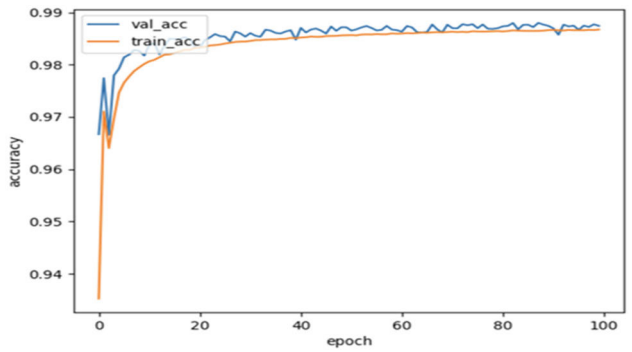


**FIGURE 11.** Accuracy values of CNN1 architecture for big_dataset with 100 epochs.

usage details for the test conducted on the big_dataset can be found in Table 11.

The test conducted on the big_dataset is optimized to nearly full capacity, with the total running time for the test amounting to 188 minutes, encompassing preprocessing and test durations. This demonstrates efficient processing for a comprehensive dataset.

Speed is indeed a critical factor for real-time protection systems. In the test, which involved 515,080 URLs from the big_dataset, the running time was a mere 3.96 seconds. This signifies that the developed system can classify a single URL in a matter of seconds. In practical terms, it can classify a remarkable 130,070 URLs in just one second, attesting to its capability for rapid, real-time protection.

### D. TESTS WITH TRADITIONAL MACHINE LEARNING ALGORITHMS

While the primary focus of this study revolved around the development of a deep learning-based phishing URL detection system, the collected dataset was also subjected to tests using traditional algorithms, and the accuracy values were compared. These tests involving traditional machine learning methods were conducted using the small dataset.

To ensure a fair and meaningful comparison between the tests, it's essential to conduct them under conditions where only the algorithms are altered as consistently as possible. Traditional machine learning algorithms have, until recently, relied on vectorization methods like n-grams for processing textual data. However, embedding, which can be derived from deep learning architectures, is increasingly being adopted for use with traditional machine learning algorithms.

**TABLE 11.** Resource utilization of GPUs.

| Resource | Utilization |
|---|---|
| CPU | 8.9% |
| Memory (59 GB) | 17.1% |
| Disk (201 GB) | 0.3% |
| GPU | 94% |
| GPU Memory (16 GB) | 97.1% |

In n-gram approaches, working with sparse data is essential, while embeddings make use of dense vectors with lower dimensions. In the domain of natural language processing, word embeddings are widely recognized and employed in literature as an effective technique. This adaptability allows for a more comprehensive evaluation of the dataset using both traditional and deep learning-based methods.

In this article, character embedding is employed instead of word embedding because the focus of the study is on vectorizing URLs using character-based operations. While pre-trained word embeddings are commonly available and used in literature, character-based embeddings are less frequently shared. Bypassing an AI model is straightforward by directly employing a predetermined word list. To safeguard the system from such factors, it was designed using character embedding.

This article specifically utilizes an embedding size of 50, as enlarging the embedding size increases the number of parameters to be trained. Therefore, the experiments did not explore varying embedding sizes, and the article focuses on the chosen embedding size of 50.

In this approach, each character is represented by a vector of size (1, 50), and character embeddings are learned during the training process. These character embeddings are drawn from the most successful deep learning architecture applied to the small_dataset, which is CNN1 trained for 100 epochs. This architecture was selected to utilize character embeddings.

The deep learning architectures investigated in this article are designed to work with sequence data. To introduce sequence data to machine learning algorithms using character embeddings, the embedding vectors for each character are employed. This allows for the effective handling and processing of sequence data within the chosen architectures.

Traditional algorithms were tested using two different vectorization methods, with the first one being concatenation. In this vectorization method, designed to match the maximum character length used in deep learning architectures (which was set to 200 characters), 50-dimensional vectors for each character were concatenated. This resulted in generating 10,000 ($200 \times 50$) dimension vectors for each data sample.

The generated vectors were then subjected to testing using three different traditional machine learning algorithms. One of the primary objectives of testing these traditional algorithms was to highlight the differences between the training processes employed in deep learning approaches and to evaluate the accuracy rates achieved by deep learning methods when applied to the same problem. This comparison allowed for an assessment of the necessity of employing deep learning techniques for solving the phishing detection problem, ultimately demonstrating that deep learning architectures were more effective in addressing this issue.

The three algorithms tested were Random Forest, Logistic Regression, and Naive Bayes, which are statistics-based algorithms. The small dataset used in these experiments contained 364,199 training samples, 104,576 validation samples, and 51,510 test samples. The performance values obtained on the small dataset using these traditional machine learning algorithms are presented in Table 12 on the validation set using the 50 and 10,000-dimensional vectors. The second vectorization method is tested with traditional algorithms involving character embeddings.

**TABLE 12. Performance metrics FOR 50/10000 dimensional vectors.**

| Algorithm | Dimen. | Precision | Recall | F1-Score | Test Acc. |
|---|---|---|---|---|---|
| Logistic Regression | 10000 | **0.94** | **0.94** | **0.94** | 0.938 |
| Naïve Bayes | 10000 | 0.66 | 0.62 | 0.56 | 0.616 |
| Random Forest | 10000 | 0.92 | 0.92 | 0.92 | 0.916 |
| Logistic Regression | 50 | 0.87 | 0.87 | 0.87 | 0.869 |
| Naïve Bayes | 50 | 0.77 | 0.72 | 0.7 | 0.726 |
| Random Forest | 50 | **0.88** | **0.88** | **0.88** | 0.878 |

In this method, each URL is represented by a 50 and 10,000-dimensional vectors calculated as the average of all character embeddings within the URL.

The average values are taken at the same index number for each character embedding. For instance, the first value of the output vector is computed as the average of the first values of all character embeddings. According to the results, the logistic regression algorithm was the most successful with the vectorization created by concatenation, achieving a validation accuracy of 0.938. On the other hand, the Random Forest algorithm was the most successful with a validation accuracy of 0.877.

The highest accuracy rate observed in the tests using deep learning was 0.987. The results of traditional machine learning algorithms were somewhat less successful than those of deep learning approaches. It's worth noting that this article primarily employed character-based vectorization, omitting word-based features from URLs, and thus, focusing on low-level features. Deep learning approaches have proven more successful in detecting phishing URLs when handling low-level features. It's speculated that traditional machine learning algorithms, which tend to emphasize high-level features, could potentially yield improved results if they could utilize word-based features. However, such an investigation is beyond the scope of this study.

### E. EVALUATION CRITERIA

The system developed in this article will analyze whether a URL is phishing or legitimate. In the case of binary classification problems, four different situations may occur in the test phase. These are the following:

- False Negative (FN): A condition in which a harmful URL is classified as clean.
- False Positive (FP): This is the case where the URL is reported as harmful even if it is not harmful. In this case, the system does not allow this access when the user wants to access a legitimate domain.
- True Negative (TN): A clean condition of clean URLs.
- True Positive (TP): This is where harmful URLs are classified as harmful.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{1}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2}$$

$$\text{F} - \text{Measure} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FN} + \text{FP}} \tag{4}$$

In case of a successful binary classification problem, it is expected that the number of samples related to TP and TN status will be higher, and the number of samples related to FP and FN status will be lower.

The metrics used for the evaluation of the test results are as follows: precision, recall, f-measure, and accuracy. These statistics, whose formulation is depicted in Equations (1)–(4), are also important for making a comparison between the tested machine learning approaches.

## VI. DISCUSSION

In this research paper, the primary objective is to create a high-performing phishing detection system using Deep Learning techniques. To achieve this goal, a substantial amount of data is required for training and testing the detection system effectively. Within the context of this study, a significant dataset was collected and made publicly available for the specific purpose of detecting phishing attacks. Notably, this dataset is one of the largest known datasets for phishing detection in academic literature.

The dataset comprises a total of 2,320,893 URLs associated with phishing attacks and 2,881,948 URLs corresponding to legitimate websites. Such a large and balanced dataset is crucial for developing and evaluating machine learning models capable of accurately distinguishing between phishing URLs and legitimate ones. This rich dataset is expected to contribute to the development of robust and high-performing

phishing detection algorithms. The evaluation of the dataset involved the utilization of deep learning algorithms, which are known for their capability to handle extensive datasets. Given the substantial volume of data, significant processing power is required to perform the necessary computations effectively.

For the convenience of conducting experiments within this study and to facilitate further research in the field, two distinct versions of the dataset were made available. The first version is a sub-sample of the dataset, comprising 10% of the complete dataset, and is referred to as the ''small_dataset.'' The second version, termed the ''big_dataset,'' encompasses the entire dataset, providing researchers with a comprehensive and larger dataset to work with. These two dataset versions are designed to accommodate different research needs and the computational resources available to researchers.

Deep learning algorithms are known for their reliance on numerous hyperparameters, which are essential for fine-tuning and optimizing the performance of these models. In this study, some of the hyperparameters were adjusted while others were maintained at constant values. The experimentation process involved the use of an Nvidia Tesla V100 GPU, which was rented for the purpose of conducting the tests. To identify suitable hyperparameters and architectural configurations, initial tests were carried out on the ''small_dataset.'' Once the optimal configuration was determined for this dataset, it was subsequently evaluated on the more extensive ''big_dataset.'' Among the various deep learning architectures that were tested, the Convolutional Neural Network (CNN) demonstrated the highest level of success, achieving an impressive accuracy rate of 98.74%.

The developed system exhibits an impressive ability to classify over 130,000 URLs per second, and this performance can be further improved by utilizing more powerful processing units. In addition to exploring deep learning architectures, traditional machine learning algorithms, such as Naïve Bayes, Random Forest, and Logistic Regression, were also evaluated. Two different vectorization methods that utilized character embeddings were employed in these assessments.

Among the traditional algorithms tested, the logistic regression algorithm emerged as the most effective, achieving a notable validation accuracy of 93.8% in our experiments. The proposed deep learning model on CNN structure can reach 98.7% accuracy value system which offers several significant advantages, as outlined below:

### A. LANGUAGE INDEPENDENCE

Many phishing detection systems are heavily dependent on language-specific features, making them less versatile. In contrast, the proposed system can detect phishing using URL information alone, focusing solely on the characters within the URL. This word-independent approach enables the system to detect phishing attacks regardless of the language used.

### B. HUGE SIZE OF PHISHING AND LEGITIMATE DATA

Constructing a dataset for an anti-phishing system is often a challenging task. To achieve high accuracy rates with machine learning algorithms, a substantial amount of training data is crucial.

### C. REAL-TIME EXECUTION

As phishing attackers can quickly create fraudulent web pages that are active for short durations, real-time detection is essential for effective prevention. The proposed system can classify over 130,070 URLs per second, making it suitable for deployment in high-traffic environments. Furthermore, this processing rate can be further improved by leveraging powerful GPUs, enhancing real-time protection capabilities.

### D. DETECTION OF NEW WEBSITES

The proposed system can identify new phishing websites that haven't been previously classified as phishing. This capability makes the system resilient against zero-day attacks, a particularly dangerous type of phishing attack. By detecting threats at the initial stages of a URL, the system significantly reduces the potential impact of such attacks.

### E. INDEPENDENCE FROM THIRD-PARTY SERVICES

While many works in the literature rely on third-party services like ''whois'' records, web-based blacklists/whitelists, and network traffic measures to enhance detection and prevention, these services can introduce delays in real-time execution. The proposed system operates independently of such third-party services, ensuring efficient and rapid phishing detection, especially in high-traffic settings.

## VII. CONCLUSION AND FUTURE WORKS

The widespread use of the Internet has increased the importance of the security of assets on the Internet. Phishing attacks are one type of cyber-attack that threatens users today. This type of attack is intended to exploit people's weaknesses. In this article, a deep learning-based system for detecting fishing attacks was proposed, and five different deep learning architectures were tested. These architectures are recurrent neural networks, bi-directional neural networks, convolutional neural networks, artificial neural networks, and attention-based networks.

The study revealed that the results obtained from deep learning approaches outperformed those of traditional machine learning algorithms. The primary focus of this research was the development of a system for phishing attack detection from URLs using deep learning methods. To facilitate transparency and access to the findings, all application codes and datasets pertaining to the deep learning tests conducted in this study have been shared through Github, IEEE DataPort and Code Ocean [38], [39], [40]. These shared resources include comprehensive information about all tests, encompassing accuracy and loss plots, confusion matrices, GPU utilization data, model and

weight files, precision, recall, f1-score metrics, and execution times. This commitment to providing detailed information ensures reproducibility and thorough analysis of the study's results.

## ACKNOWLEDGMENT

The authors would like to thank Roksit for their support during the implementation of this work.

## REFERENCES

[1] Anti-Phishing Working Group. (Sep. 2022). *Phishing Attacks Trends Report-Q2 2022*. Accessed: Oct. 15, 2022. [Online]. Available: https://apwg.org/trendsreports/

[2] *Cloudflare's 2023 Phishing Threats Report*. Accessed: Oct. 1, 2023. [Online]. Available: https://www.cloudflare.com/lp/2023-phishing-report/

[3] M. Volkamer, K. Renaud, B. Reinheimer, and A. Kunz, "User experiences of TORPEDO: Tooltip-powered phishing email detection," *Comput. Secur.*, vol. 71, pp. 100–113, Nov. 2017, doi: 10.1016/j.cose.2017.02.004.

[4] N. Q. Do, A. Selamat, O. Krejcar, E. Herrera-Viedma, and H. Fujita, "Deep learning for phishing detection: Taxonomy, current challenges and future directions," *IEEE Access*, vol. 10, pp. 36429–36463, 2022, doi: 10.1109/ACCESS.2022.3151903.

[5] T. Mahara, V. L. H. Josephine, R. Srinivasan, P. Prakash, A. D. Algarni, and O. P. Verma, "Deep vs. shallow: A comparative study of machine learning and deep learning approaches for fake health news detection," *IEEE Access*, vol. 11, pp. 79330–79340, 2023, doi: 10.1109/ACCESS.2023.3298441.

[6] *Google Safe Browsing*. Accessed: Oct. 1, 2023. [Online]. Available: https://safebrowsing.google.com/

[7] (2019). *Office 365 Advanced Threat Protection Safe Links*. Accessed: Jul. 10, 2023. [Online]. Available: https://docs.microsoft.com/en-us/office365/securitycompliance/atp-safe-links

[8] 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, Dec. 2016, doi: 10.1186/s13635-016-0034-3.

[9] N. A. Azeez, S. Misra, I. A. Margaret, L. Fernandez-Sanz, and S. M. Abdulhamid, "Adopting automated whitelist approach for detecting phishing attacks," *Comput. Secur.*, vol. 108, Sep. 2021, Art. no. 102328, doi: 10.1016/j.cose.2021.102328.

[10] N. Abdelhamid, A. Ayesh, and F. Thabtah, "Phishing detection based associative classification data mining," *Expert Syst. Appl.*, vol. 41, no. 13, pp. 5948–5959, Oct. 2014, doi: 10.1016/j.eswa.2014.03.019.

[11] M. Moghimi and A. Y. Varjani, "New rule-based phishing detection method," *Expert Syst. Appl.*, vol. 53, pp. 231–242, Jul. 2016, doi: 10.1016/j.eswa.2016.01.028.

[12] M. SatheeshKumar, K. G. Srinivasagan, and G. UnniKrishnan, "A lightweight and proactive rule-based incremental construction approach to detect phishing scam," *Inf. Technol. Manage.*, vol. 23, no. 4, pp. 271–298, Dec. 2022, doi: 10.1007/s10799-021-00351-7.

[13] A. K. Jain and B. B. Gupta, "Phishing detection: Analysis of visual similarity based approaches," *Secur. Commun. Netw.*, vol. 2017, pp. 1–20, Oct. 2017, doi: 10.1155/2017/5421046.

[14] E. Medvet, E. Kirda, and C. Kruegel, "Visual-similarity-based phishing detection," in *Proc. 4th Int. Conf. Secur. Privacy Commun. Netowrk*, Sep. 2008, pp. 1–6, doi: 10.1145/1460877.1460905.

[15] W. Liu, X. Deng, G. Huang, and A. Y. Fu, "An antiphishing strategy based on visual similarity assessment," *IEEE Internet Comput.*, vol. 10, no. 2, pp. 58–65, Mar. 2006, doi: 10.1109/MIC.2006.23.

[16] Y. Zhou, Y. Zhang, J. Xiao, Y. Wang, and W. Lin, "Visual similarity based anti-phishing with the combination of local and global features," in *Proc. IEEE 13th Int. Conf. Trust, Secur. Privacy Comput. Commun.*, Sep. 2014, pp. 189–196, doi: 10.1109/TRUSTCOM.2014.28.

[17] G. Varshney, M. Misra, and P. K. Atrey, "Improving the accuracy of search engine based anti-phishing solutions using lightweight features," in *Proc. 11th Int. Conf. Internet Technol. Secured Trans. (ICITST)*, Dec. 2016, pp. 365–370, doi: 10.1109/ICITST.2016.7856731.

[18] P. A. Watters, A. Herps, R. Layton, and S. McCombie, "ICANN or ICANT: Is WHOIS an enabler of cybercrime?" in *Proc. 4th Cybercrime Trustworthy Comput. Workshop*, Nov. 2013, pp. 44–49, doi: 10.1109/CTC.2013.13.

[19] H. Kim and J. H. Huh, "Detecting DNS-poisoning-based phishing attacks from their network performance characteristics," *Electron. Lett.*, vol. 47, no. 11, p. 656, 2011, doi: 10.1049/el.2011.0399.

[20] A. Karim, M. Shahroz, K. Mustofa, S. B. Belhaouari, and S. R. K. Joga, "Phishing detection system through hybrid machine learning based on URL," *IEEE Access*, vol. 11, pp. 36805–36822, 2023, doi: 10.1109/ACCESS.2023.3252366.

[21] M. K. Prabakaran, P. Meenakshi Sundaram, and A. D. Chandrasekar, "An enhanced deep learning-based phishing detection mechanism to effectively identify malicious URLs using variational autoencoders," *IET Inf. Secur.*, vol. 17, no. 3, pp. 423–440, May 2023, doi: 10.1049/ise2.12106.

[22] Z. Alshingiti, R. Alaqel, J. Al-Muhtadi, Q. E. U. Haq, K. Saleem, and M. H. Faheem, "A deep learning-based phishing detection system using CNN, LSTM, and LSTM-CNN," *Electronics*, vol. 12, no. 1, p. 232, Jan. 2023, doi: 10.3390/electronics12010232.

[23] M. Sánchez-Paniagua, E. F. Fernández, E. Alegre, W. Al-Nabki, and V. González-Castro, "Phishing URL detection: A real-case scenario through login URLs," *IEEE Access*, vol. 10, pp. 42949–42960, 2022, doi: 10.1109/ACCESS.2022.3168681.

[24] S. He, B. Li, H. Peng, J. Xin, and E. Zhang, "An effective cost-sensitive XGBoost method for malicious URLs detection in imbalanced dataset," *IEEE Access*, vol. 9, pp. 93089–93096, 2021, doi: 10.1109/ACCESS.2021.3093094.

[25] S. Tiwari, H. Rizvi, and K. Kalaiselvi, "Malicious website navigation prevention using CNNs and URL vectors: A study," in *Proc. Int. Conf. Comput. Commun. Informat. (ICCCI)*, Jan. 2022, pp. 1–6, doi: 10.1109/ICCCI54379.2022.9741056.

[26] T. Rasymas and L. Dovydaitis, "Detection of phishing URLs by using deep learning approach and multiple features combinations," *Baltic J. Modern Comput.*, vol. 8, no. 3, pp. 471–483, Sep. 2020, doi: 10.22364/bjmc.2020.8.3.06.

[27] T. Feng and C. Yue, "Visualizing and interpreting RNN models in URL-based phishing detection," in *Proc. 25th ACM Symp. Access Control Models Technol.*, Jun. 2020, pp. 13–24, doi: 10.1145/3381991.3395602.

[28] A. Aljofey, Q. Jiang, Q. Qu, M. Huang, and J.-P. Niyigena, "An effective phishing detection model based on character level convolutional neural network from URL," *Electronics*, vol. 9, no. 9, p. 1514, Sep. 2020, doi: 10.3390/electronics9091514.

[29] Y. Huang, Q. Yang, J. Qin, and W. Wen, "Phishing URL detection via CNN and attention-based hierarchical RNN," in *Proc. 18th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun./13th IEEE Int. Conf. Big Data Sci. Eng.*, Aug. 2019, pp. 112–119, doi: 10.1109/TrustCom/BIGDATASE.2019.00024.

[30] 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, doi: 10.1016/j.eswa.2018.09.029.

[31] S. Singh, M. P. Singh, and R. Pandey, "Phishing detection from URLs using deep learning approach," in *Proc. 5th Int. Conf. Comput., Commun. Secur. (ICCCS)*, Oct. 2020, pp. 1–4, doi: 10.1109/ICCCS49678.2020.9277459.

[32] L. Tang and Q. H. Mahmoud, "A deep learning-based framework for phishing website detection," *IEEE Access*, vol. 10, pp. 1509–1521, 2022, doi: 10.1109/ACCESS.2021.3137636.

[33] J. Anitha and M. Kalaiarasu, "A new hybrid deep learning-based phishing detection system using MCS-DNN classifier," *Neural Comput. Appl.*, vol. 34, no. 8, pp. 5867–5882, Apr. 2022, doi: 10.1007/s00521-021-06717-w.

[34] S. Das Guptta, K. T. Shahriar, H. Alqahtani, D. Alsalman, and I. H. Sarker, "Modeling hybrid feature-based phishing websites detection using machine learning techniques," *Ann. Data Sci.*, vol. 2022, pp. 1–26, Mar. 2022, doi: 10.1007/s40745-022-00379-8.

[35] M. Alshehri, A. Abugabah, A. Algarni, and S. Almotairi, "Character-level word encoding deep learning model for combating cyber threats in phishing URL detection," *Comput. Electr. Eng.*, vol. 100, Mar. 2022, Art. no. 107868, doi: 10.1016/j.compeleceng.2022.107868.

[36] A. Mughaid, S. AlZu'bi, A. Hnaif, S. Taamneh, A. Alnajjar, and E. A. Elsoud, "An intelligent cyber security phishing detection system using deep learning techniques," *Cluster Comput.*, vol. 25, no. 6, pp. 3819–3828, Dec. 2022, doi: 10.1007/s10586-022-03604-4.
[37] Z.-H. Zhan, J.-Y. Li, and J. Zhang, "Evolutionary deep learning: A survey," *Neurocomputing*, vol. 483, pp. 42–58, Apr. 2022, doi: 10.1016/j.neucom.2022.01.099.
[38] *Github Resources*. Accessed: Dec. 3, 2023. [Online]. Available: https://github.com/ebubekirbbr/phishing_url_detection
[39] *Deep Learning Based Phishing Detection System (DEPHIDES)*. Accessed: Dec. 3, 2023. [Online]. Available: https://codeocean.com/capsule/0874584/tree
[40] *Phishing Attack Dataset*. Accessed: Dec. 3, 2023. [Online]. Available: https://dx.doi.org/10.21227/4098-8c60

**OZGUR KORAY SAHINGOZ** received the B.Sc. degree from the Computer Engineering Department, Boğaziçi University, in 1993, and the M.S. and Ph.D. degrees from the Computer Engineering Department, Istanbul Technical University, in 1998 and 2006, respectively. He is currently a Professor with the Computer Engineering Department, Biruni University, İstanbul. His research interests include artificial intelligence, machine/deep learning, data science, and UAV networking.

**EBUBEKIR BUBER** received the bachelor's, graduate, and Ph.D. degrees in artificial intelligence, and the master's degree in cyber security.

He was the Manager in projects involving artificial intelligence in the field of cyber security during his business experience. To develop more efficient projects in this field, he received the master's degree. For over seven years, he has been working on artificial intelligence projects in the field of cyber security. He has also created several academic publications on AI and cyber security.

**EMIN KUGU** (Member, IEEE) received the B.S. degree from the Computer Science Engineering Department, Istanbul University, the master's degree in software engineering from Air Force Academy, and the Ph.D. degree in electrical and computer engineering program from Old Dominion University. He was a part-time Lecturer with different universities. He was the Project Manager in the Air Force Command Information System (HvBS) Project and took part in the software development processes of international defense projects. His research interests include cyber security, computer networks, artificial intelligence, and machine/deep learning.