Received 15 May 2024, accepted 29 May 2024, date of publication 3 June 2024, date of current version 11 June 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3409049

RESEARCH ARTICLE

An Effective Detection Approach for Phishing URL Using ResMLP

S. REMYA[1], MANU J. PILLAI[2], KAJAL K. NAIR[2], SOMULA RAMA SUBBAREDDY[3], AND YONG YUN CHO[4]

[1]Amrita School of Computing, Amrita Vishwa Vidyapeetham, Amritapuri, Kollam, Kerala 690525, India
[2]Department of Computer Science and Engineering, TKM College of Engineering, Kollam, Kerala 691005, India
[3]Department of Information Technology, Vallurupalli Nageswara Rao Vignana Jyothi Institute of Engineering and Technology, Hyderabad, Telangana 500090, India
[4]Department of Information and Communication Engineering, Sunchon National University, Jeollanam-do, Suncheon 57922, South Korea

Corresponding author: Yong Yun Cho (yycho@scnu.ac.kr)

This work was supported in part by the Innovative Human Resource Development for Local Intellectualization Program through the Institute of Information and Communications Technology Planning and Evaluation (IITP) Grant funded by Korean Government, Ministry of Science and ICT (MSIT), South Korea, under Grant IITP-2024-2020-0-01489, 50%; and in part by the MSIT through the Information Technology Research Center (ITRC) Support Program supervised by IITP under Grant RS-2024-00259703, 50%.

ABSTRACT Phishing websites, mimicking legitimate counterparts, pose significant threats by stealing user information through deceptive Uniform Resource Locators (URLs). Traditional blacklists struggle to identify dynamic URLs, necessitating advanced detection mechanisms. In this study, we propose an effective approach utilizing residual pipelining for phishing URL detection. Our method extracts common URL features and sentiments, employing a residual pipeline comprising convolutional and inverted residual blocks. These resultant features are then fed into a Multi-Layer Perceptron (MLP) for classification. We evaluate the efficacy of our approach against traditional algorithms using a Kaggle dataset. Our results demonstrate superior accuracy, precision, F1 Score, and recall, showcasing its effectiveness in mitigating phishing threats. Utilizing a residual pipeline made up of convolutional and inverted residual blocks, we start our method by identifying similar URL features and sentiments. We also use domain age research to figure out how long URLs have been around. Additionally, the lexical study of URL structure makes our method more useful, resulting in impressive accuracy. With an accuracy of 98.29%, this research highlights the importance of innovative techniques in combating evolving cyber threats. Future research directions could focus on enhancing the model's robustness against adversarial attacks and integrating real-time monitoring for proactive defense strategies.

INDEX TERMS Phishing, URL detection, residual pipelining, cybersecurity, classification.

I. INTRODUCTION

Phishing websites are malicious websites that look similar to legitimate websites in terms of their web pages and Uniform Resource Locator(URL) addresses. Phishing takes the form of URL phishing, in which a threat actor manipulates internet URLs in a variety of ways to encourage their targets to click them. Usually, clicking on these links leads individuals to fraudulent, malware-infected websites that look for sensitive personal data, such as banking account details and passwords. The victims of these connections may suffer severe consequences [1]. Realistically, detecting and recognizing phishing websites is a dynamic and difficult endeavor. Phishing offenses can be conducted through several channels, including via email, websites, spyware, SMS, and voice calls. Among the most prominent types of URL phishing attacks is, when a fraudster impersonates a well-known company and sends a bogus email with the message "Your account has been disabled". In response, alarmed users click the link, unknowingly downloading malware onto their computers.

Phishing attempts have risen by 33% annually since 2015 on average. Due to the expansion of the internet and the number of people working from home, phishing

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

has increased more than twice as much as it did in 2022. December 2021 witnessed 316,747 attacks, reported by the Anti-Phishing Working Group's (APWG) [2], the highest case in its history. Bank-related phishing assaults accounted for 23.2% of all phishing attacks in the fourth quarter of 2021, according to OpSec Security, a founding member of APWG.

Anti-phishing is the technique of preventing phishing attacks in which attackers try to get sensitive information through non-repudiation. Attacker's tactics and methods of targeting have advanced significantly as common phishing techniques have become more transparent to the general public. Many businesses have created anti-phishing systems [3] to reduce these hazards, however, these tools are not the last layer of defense. Anti-phishing software is a platform or series of software services that can identify malicious inbound messages that pose as authentic or try to gain trust through social engineering. It also allows users to create whitelists and blacklists for message filtering and takes preventative measures when necessary [4]. However, these are insufficient to battle phishing, since attackers make use of one-time phishing URLs. Machine learning techniques are used to deal with this trick, depending on an integrated classifier to look at the properties of sample URLs [5], [6] to make judgments for new, developing ones[21]. Likewise, deep learning-based methods [8] are developed which are capable of classifying data more accurately than traditional ML models.

A novel approach is introduced in this research work leveraging residual pipelining methodology to enhance the efficacy of traditional detection mechanisms, where URL features along with a few sentimental features are collected as part of feature extraction and transformed into a matrix. Then this matrix is fed onto the residual pipeline module which consists of convolution layers and inverted residual layers. After that, the obtained result is fed into an output block where the actual classification of the URL takes place. In response to the escalating sophistication of phishing attacks, we provide an effective solution using a hybrid feature set. This collection includes different hyperlink information, and URL character sequence characteristics, culminating in the creation of feature vectors necessary for training our anti-phishing model. Ultimately, rigorous testing reveals that the accuracy reaches up to 98.295%, which outperforms the conventional methods.

Our anti-phishing solution is meticulously designed to fulfill stringent requirements essential for robust detection and prevention of phishing attacks. It prioritizes high detection efficiency, real-time detection capabilities, target independence, and third-party independence. By minimizing false positives and maximizing true positives, our method ensures timely prediction of phishing attempts while maintaining adaptability to emerging threats without reliance on external services.

Key contributions of our research include:

- The proposal of a phishing detection approach that integrates residual pipelining methodology, offering enhanced performance and resilience against evolving cyber threats.
- Additionally, we introduce a comprehensive feature set comprising novel and existing features, boosting the effectiveness of our detection mechanism.
- Through extensive experimentation and evaluation, we validate the precision and accuracy of our method in identifying legitimate websites while minimizing false positive rates.

To present our research findings logically, the subsequent sections are organized as follows. The section on Literature Review thoroughly examines relevant publications that form the basis of our research and offer insights into current phishing detection techniques and approaches. The methodology section provides comprehensive details about the technical principles and theoretical foundations of our suggested methodology, which are crucial for placing our research in context. The dataset that we used for our experiments is presented in the section Dataset and Experimental Results, along with a detailed analysis of the findings of our research. In the result section, we also provide in-depth analyses of the performance of our proposed approach across various evaluation metrics. Concisely summarising our results, the conclusion section explores future directions for phishing detection research and development.

II. RELATED WORKS

Phishing has long been one of the most popular cyber-attack strategies used by bad actors. The problems posed by phishing websites have been addressed by numerous studies. Many techniques for detecting phishing websites have been proposed, including blacklist-based and heuristic-based techniques. The statistics from the training dataset have a substantial impact on the weights in the heuristic-based approach. Blacklists [9], which is a dataset consisting of malicious URLs are still used by several internet companies. However, it is unable to forecast outcomes for a new URL that has not yet been added to the list, because attackers are increasingly using one-time URLs to carry out attacks. To address this issue several approaches have been developed.

Xiao et al [10] developed CNN-MHSA, a combination of Convolutional Neural Network (CNN) and multi-head self-attention mechanisms to detect phishing websites. In this method, feature extraction and weight calculation are performed independently by duplicating the input matrix into two. The self-attention mechanism then aids in identifying whether websites are malicious or benign. This method exhibits strong performance in differentiating phishing websites from authentic ones by utilizing CNN's capability for spatial feature learning and self-attention for collecting long-range relationships. Model CNN-MHSA, improved efficiency and interpretability can be seen in its ability to decouple the weight calculation procedure from feature extraction. It is crucial to recognize that, even with the encouraging results, the complex neural network architecture may need a significant amount of computing power for both training and inference. The efficacy of the model can also be impacted by factors such as the variety and quality of the training data as well as the dynamic tactics employed by phishing opponents.

For precise phishing detection, Weiping Wang et al. [11] established a method called Recurrent Convolutional Neural Networks (PDRCNN). A two-dimensional tensor representation generated by the PDRCNN is given as an input for classification purposes. By utilizing these attributes, the model can identify temporal and spatial patterns of URL data, which improves the identification of phishing efforts. Large labeled datasets are necessary, high computational resource needs, and overfitting vulnerability are some of the drawbacks of PDRCNN, despite its advantages such as its ability to handle sequential data and adapt to various URL formats. PDRCNN cannot be successfully used in actual cybersecurity applications until these problems are fixed.

The Phishing Hybrid Feature-Based Classifier (PHFBC), designed by Zuhair et al. [12], combines recursive feature subset selection with ML approaches to produce a comprehensive phishing detection system. With a set of features gathered from phishing and legitimate websites, their objective was to accurately classify phishing. PHFBC incorporates decision tree and Naive Bayes models using a statistical measure known as the Phish Ratio. Though it is an innovative technique, it has limitations such as being susceptible to feature replication, requiring laborious and prone to error manual feature extraction, and having trouble selecting the best features for different phishing scenarios. Furthermore, parameters like representativeness in response to changing phishing strategies could have an impact on how successful PHFBC is. Resolving these issues would improve PHFBC's resilience and suitability for use in actual phishing detection situations.

Ramesh et al. [13] provided a technique for identifying phishing webpages and their target domains through simulation analysis. Utilizing row and column sums, the technique determines target linkages, producing a parasitic matrix that depicts the connection between two sites. However, scalability problems with this method could appear when handling huge datasets or intricate website architecture. Additionally, how well the technique works may depend on the accuracy of the presumed correlations discovered and the consistency of the row and column total computations. The human-generated parasite matrix may also produce biased or erroneous results, and the system may not be able to adapt to evolving phishing or website design trends.

Cova et al. [14] intended to comprehend the basic design and application of phishing kits to determine the methods of deception used by hackers to hide backdoors they had installed and to educate interested parties about the techniques that phishers normally use to send phished data. Although their research offers insightful information about the strategies and methods used in phishing attempts, there might be some restrictions on it. For example, the availability and variety of phishing kits evaluated, as well as the timeliness and accuracy of the data acquired, could limit the efficacy. Furthermore, the study generalizes the findings to more extensive phishing threats and attack scenarios. Apart from that, it might be difficult to analyze the dynamic nature of phishing attempts and the quick development of phishing strategies. Notwithstanding these possible drawbacks, this investigation adds a great deal to our knowledge of how phishing kits are used and develop stronger defences against phishing attacks.

The novel technique "Antiphishing through Phishing Target Discovery," by Liu et al. [15], aims to detect possible phishing websites through an analysis of their parasitic community structure. Identifying the principal phishing target webpage and exposing "parasitic" connections are the goals of the above method, which collects webpages that are either directly or indirectly linked to a certain suspicious webpage. However there is a chance that this approach may fail, especially in cases when the parasitic community structure is dynamic or complex. Furthermore, the quality and completeness of the web link data used for analysis, as well as the variety of phishing strategies and techniques used by the attackers, could have an impact on the accuracy of the methodology.

CANTINA is a content-based method developed by Zhang et al. [16] that analyses character scores and extracts keywords from website texts to identify phishing websites. Notwithstanding its inventive methodology, CANTINA can encounter various constraints. For example, the use of TF-IDF analysis and character scores alone may miss less obvious signs of phishing activity, like visual cues or contextual components. In addition, the model's efficacy might be restricted by the level of accuracy and significance of the selected keywords in addition to the possibility of false positives or false negatives in the Google search results. Furthermore, Google search rankings as a metric for legitimacy could lead to bias and inaccurate results, especially when phishing websites alter search engine results or genuine websites are not highly ranked.

To create a classification model, Marchal et al. [17] used a feature extraction technique, extracting 212 features and applying Gradient Boosting. Although this methodology is a thorough attempt to capture several aspects of phishing websites, it might run into issues with feature selection and model complexity. The amount of features that are extracted may cause problems like overfitting, particularly if some of the features are irrelevant to the purpose of phishing detection. Furthermore, the process of manually extracting features can be time-consuming and may eliminate important details from phishing websites, which could reduce the ability of the model to identify new and developing phishing techniques. Furthermore, considering large-scale deployment scenarios when computational resources are limited, the selection of Gradient Boosting as the classification algorithm

TABLE 1. Literature review of phishing detection methods.

Author/Title	Description	Dataset	Limitations

Xi Xiao et al.	Proposed a combination of CNN and multi-head self-attention
	Features are extracted and weights calculated separately
	Self-attention is employed for classification
		Private dataset
		Benign(45,000)
		Phishing(43,984)
			The robustness of the model may be affected by the URL length parameter.

Weiping Wang et al.	Presented PDRCNN, a model using Recurrent Convolutional Neural Networks to encode URL information into a two-dimensional tensor for classification.	Public dataset
		Benign(30,649)
		Phishing(29,496)
			Not effective when the webpage contains a few number of hyperlinks

Zuhair, H. et al.	Developed a hybrid feature based classifier using recursive features Subset Selection and ML algorithms.
	Features are manually extracted and hybridized for comprehensive phishing characterization.
		Public dataset
		Benign(5438)
		Phishing(4097)
			Manual feature extraction may limit scalability and adaptability. The benign websites are always subjected to two-stage verification.

Ramesh et al.	Identifying phishing webpages and their target domains by analyzing their semantics. Constructs a parasitic matrix to visualize site relations and identifies target links using row/column sums.	Public dataset
		Benign(85,409)
		Phishing(40,668)
			May not effectively detect sophisticated phishing techniques beyond simple relationships. Depends on the URL of the website.

Cova et al.	Conducted an analysis to understand phishing kit layouts and obfuscation strategies. Aims to locate and alert interested parties about phisher methods.	Private dataset
		Benign(157,626)
		Phishing(161,016)
			Focuses primarily on understanding phishing methods rather than detection.

Liu, W. et al.	Introduced Antiphishing through Phishing Target Discovery, a method to identify phishing targets by analyzing site linking patterns. Identifies parasitic communities and phishing targets based on link connections.	Private dataset
		Benign(344,794)
		Phishing(71,556)
			Relies heavily on web linking patterns, may not capture subtle phishing attempts.

Zhang, Y. et al.	Designed CANTINA, a content-based approach to detect phishing websites. Utilizes TF-IDF to score characters and extract keywords for classification. Performs Google searches and validates legitimacy based on domain appearance.	Private dataset
		Benign(1918)
		Phishing(2141)
			Dependency on Google searches may limit real-time applicability and scalability.

Marchal et al.	Extracted 212 features from URL and HTML, used Gradient Boosting for classification. Employs ML algorithms to determine phishing authenticity based on extracted characteristics.	Public dataset
		Benign(5076)
		Phishing(5438)
			Manual feature extraction may limit scalability and adaptability to evolving phishing techniques.

Mohammad et al.	Proposed Predicting Phishing Websites based on self-structuring neural networks. Developed an ANN-based model with high fault tolerance for predicting phishing attacks.	Public dataset
		Benign(5076)
		Phishing(5438)
			Complexity of neural network architecture may limit interpretability and scalability.

Nguyen et al.	Introduced An Efficient Approach for Phishing Detection using single-layer neural networks. Calculates heuristics and generates weights using a single-layer neural network.	Public data set
		Benign(32,972)
		Phishing(27,280)
			Reliance on heuristics may limit adaptability to evolving phishing techniques.

Zhang and Li	Proposed Phishing Detection Method Based on Borderline-SMOTE Deep Belief Network. Utilizes Borderline-SMOTE DBN to identify phishing using calculated probability distribution.	Public dataset
		Benign(5076)
		Phishing(5438)
			Dependency on calculated probability distribution may limit adaptability to dynamic phishing attempts.

Verma et al.	online learning with n-gram for phishing detection. Splits URLs into n-grams and uses various online learning algorithms for detection.	Private dataset
		Benign(344,794)
		Phishing(71,556)
			May not capture contextual information crucial for detecting sophisticated phishing techniques.

Yang et al.	Developed multidimensional features driven by Deep Learning. Utilizes deep learning for character sequence feature extraction and classification.	Public dataset
		Benign(36,400)
		Phishing(37,175)
			May not effectively capture nuanced features essential for detecting sophisticated phishing attempts.

Sun et al.	Introduced a graph-based approach for phishing website detection using graph neural networks. The method leverages the structural information of URLs and their relationships to improve detection accuracy.	Public dataset:
		Benign(344,794)
		Phishing(71,556)
			Overcomes limitations of traditional feature-based methods by capturing complex relationships among URLs.

TABLE 1. *(Continued.)* **Literature review of phishing detection methods.**

Author/Title	Description	Dataset	Limitations
Chen et al.	Proposed a deep transfer learning framework for phishing detection, transferring knowledge from pre-trained models to adapt to new datasets with limited labeled data. The method achieves high detection accuracy even with small training datasets.	Public dataset: Benign(1918) Phishing(2141)	Addresses the challenge of data scarcity in phishing detection, enabling effective detection with limited labeled examples.
Asiri et al.	Developed a real-time phishing detection system using deep reinforcement learning. The system continuously learns from user interactions with URLs to adaptively improve detection performance over time.	Real-world user interaction data	Provides a proactive defense against evolving phishing attacks by leveraging real-time learning and adaptation.

may provide interpretability and scalability issues for the model.

Self-structuring neural networks are the basis of an inventive technique proposed by Mohammad et al. [18] for comprehending phishing websites. For all its advantages—such as a self-organized neural network and a high level of noise tolerance—this method may have a few disadvantages. Neural network complexity can be a disadvantage since it can lead to problems with the interpretability of the models and processing performance. The effectiveness may also be influenced by the training data, given the dynamic and ever-changing nature of phishing attacks. Furthermore, the diverse and representative nature of the training dataset, along with the neural network's capacity to generalize across various phishing scenarios and attack vectors, could have an impact on the model's performance. Addressing these limitations will be crucial for ensuring the practical utility and effectiveness of the proposed approach.

Nguyen et al. [19] developed a single-layer neural network, which computes heuristics and generates weights using the network. This is an effective technique for phishing detection. The single-layer architecture of this approach makes it simple and computationally efficient, but it might have drawbacks. For example, the single-layer neural network's ability to identify complex patterns and relationships in the data may limit the efficacy. Additionally, in situations when the underlying data distribution is extremely complex or unpredictable, relying solely on heuristics for feature extraction and weight computation may result in less-than-ideal performance. Additionally, single-layer neural networks' lack of depth and limited representational capacity may limit the capacity to generalize, which could impair their effectiveness in phishing cases that have yet to be discovered or developed.

Zhang and Li [20] introduced Borderline-SMOTE Deep Belief Network (DBN). Improved detection accuracy and model robustness are two potential advantages of this approach, but it may also have some disadvantages. An example of this would be the representativeness and the training data quality, which could affect the effectiveness, particularly considering the challenges that imbalanced datasets in phishing detection tasks may provide. Nevertheless, the computational complexity of training deep neural networks like DBN and the additional overhead caused by oversampling techniques like Borderline-SMOTE could threaten the scalability and efficiency of the model, particularly in real-time or resource-constrained contexts. Addressing these limitations will be essential for ensuring the practical viability and effectiveness of the proposed phishing detection method in real-world cybersecurity applications.

Leveraging online learning with n-grams as a technique for phishing website identification was proposed by Verma and Das [21]. They divide URLs into n-grams to detect phishing websites. Even while this strategy has advantages, like its adaptability to new phishing strategies and its capacity to manage flowing data, it may also have disadvantages. Other variables could impact the outcome, including the choice of n-grams, the level of information in the feature representation, and the effectiveness of online learning algorithms in processing massive amounts of data quickly. The method may also not work as well or last as long if it relies too much on online training, which can lead to problems with model shifting and idea development over time.

Through deep learning-based multidimensional features, Yang et al. [22] suggested an approach to find fake websites. Because they use deep learning to identify features related to character sequences from URLs, their method makes it possible to quickly group things into categories. Dimensionality reduction, pattern recognition, and one-hot encoding and embedding of URLs are used in this method to try to find complicated patterns that point to phishing activities.

For finding fake websites, Sun et al. [23] proposed a new method using graph neural networks. Unlike traditional feature-based systems, the one created by Sun et al. does an excellent task of capturing the complex relationships between URLs. The network architecture is extensively looked at and subtle patterns linked to phishing are found using a graph neural network. Utilizing the framework of information found in URLs and the intricate links between them, their method greatly enhances the accuracy of detection. There will be significant advantages over current feature engineering methods if this new method can regularly and accurately spot complex phishing attempts.

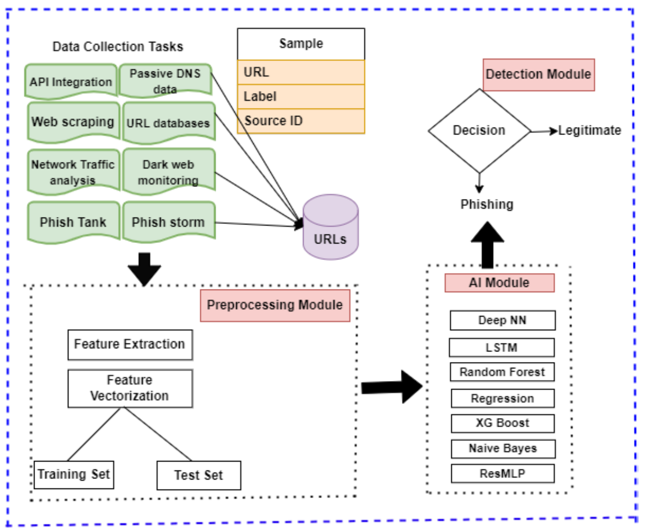


FIGURE 1. Architecture of the proposed model.

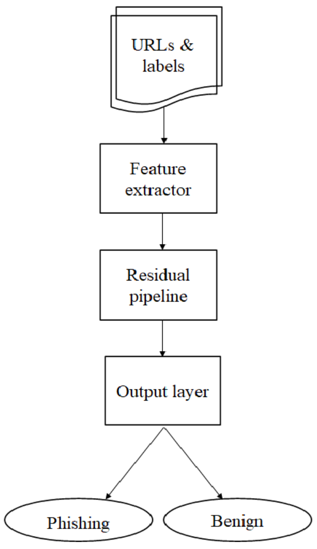


FIGURE 2. Model overview.

In order to deal with the problem of not having adequate information, Chen et al. [24] suggested a deep transfer learning system that would be optimized for finding phishing emails. Their method works well with new datasets that don't have a lot of labeled data because it uses models that have already been trained and transfer knowledge from high-quality datasets. To get high recognition accuracy with minimal training data, this method works well for generalizing models, which is helpful when it's hard to get cases that have been labeled. Their method combines domain knowledge with transferable information. Therefore, it will be possible to make detection systems that are more flexible and reliable.

Asiri et al. [25] came up with a new way to use deep reinforcement learning to find hacking attempts in real time so that security can be proactive against attempts that change all the time. Because it is always changing and learning from how people use URLs, their system gets better at finding things over time. An flexible learning method is used to make the model quickly respond to new phishing threats as they appear. This is done by using real-world data about how users interact with the system. Their system transforms into a strong defense that can find and stop phishing attempts very well through a routine of observation, action, and reward. Flexible cybersecurity solutions can start a new era with this way. High-level security is provided by these solutions, which can adapt quickly to changing cyber threat situations.

These strategies may have problems despite their benefits, such as being able to naturally learn hierarchical representations from raw data and capturing complex correlations between attributes. The efficacy of the model may be affected, for instance, by the quantity of labeled training data as well as the computational resources needed to train deep learning models. Practical application in real-world settings may also be hampered by the interpretability of the learned representations and the approach's scalability to handle large-scale datasets. It is important to tackle these constraints to guarantee the dependability and effectiveness of the proposed phishing detection technique in practical cybersecurity implementations. The summary of the existing state of the art is shown in Table 1.

III. METHODOLOGY

With the increasing threat of phishing attacks, our research aims to create a strong method for spotting phishing URLs. Central to our approach is the integration of MLP within a residual pipelining framework. This innovative amalgamation of methodologies aims to capitalize on the strengths of each approach, thereby enhancing the efficacy and accuracy of phishing website detection.

A. SYSTEM ARCHITECTURE

The overall architecture depicted in Figure 1 of the proposed approach is divided into 4 phases such as, the features are extracted in Phase 1. Feature vectorization to create a unique feature vector for every webpage is incorporated in Phase 2. Phase 3 is doing the ML part. Whether the provided webpage is phishing or not is determined in Phase 4.

1) FEATURE EXTRACTION

An integral part of our methodology is the feature extraction procedure from URLs, which is a critical step in the detection pipeline. We carefully choose and extract emotive attributes from 25 different URLs to use as input features in our detection model. A few instances of the numerous variables covered by these characteristics are the length of the URL, the host, the directory, the TLD, and the number of special characters like @, -,., =, and? Moreover, we recognize that affective dimensions are important in determining the legitimacy of URLs and account for them by considering variables such as domain age, domain registration duration, and Google indexing status. Table 2 displays the features that

TABLE 2. Features extracted from URLs.

Feature	Description
URL length	Number of characters used in the URL and determines how long it is.
Host Length	Length of the domain name in the URL
Directory length	Length of the folder or directory in the URL
TLD Length	The top-level domain (TLD) is the final segment of a domain name, following the final dot.
@count	Count of "@" symbol in the entire URL, returns the count if @ symbols are present, else returns -1.
−count	Count of − symbol in URL, -1 is returned if there is no such symbol, else count is outputted.
. count	Count of dots present in URL, returns the count of dot symbol if present, else returns -1
= count	Count of = symbol in URL, returns the count of = symbol if present, else returns -1
? count	Count of ? in the URL, returns -1 if there is no such symbol, else returns the count of that symbol
% count	Count of % symbol in URL, returns -1 if there is no such symbol, else returns the count
Hyperlink count	Count of HTTP URL and HTTPS URL
Digit count	The count of digits present in an URL, returns -1 if there is no digits, else returns the count
Letter count	Count of letters present in an URL, returns the count if letters are present, else returns -1
Directory Count	Count of folders, three directories at most are advised
Shortening service	In order to create shorter aliases for lengthy URLs, URL shortening services like bit.ly and TinyURL are quite popular [29]. If these are present then returns 1 else returns -1
Having redirect URL	Checks if the URL is having double slash redirection, if the redirection is more than 6 then returns -1, else returns 1.
Having suffix URL	To make a phishing website appear legitimate, a prefix or suffix separated by a hyphen can be added. Returns -1 is no such URL is present, else returns 1.
Having subdomain	A subdomain is the portion of a URL that is to the left of the domain name. If its present then returns 1, else returns -1
DNS	The name of the website, or URL, and the specific IP address it connects to are maintained and mapped by Domain Name System (DNS). Every URL on the internet is uniquely identified by its IP address, which is that of the machine hosting the website's server.
Having IP address	Having IP address - Checks if the host or domain part of an URL is IP
Domain Registration Length	Number of days since the domain was registered as of today
HTTP Token	HTTP token in the URL
Is Abnormal URL	Checks if there is any violation from regular pattern of URL
Age Of Domain	Maintain track of how long your website has been online
Google Index	Verify a URL's Google indexing status

we have carefully chosen to capture the nuances of patterns and characteristics present in phishing URLs. As a result, our detection system gained a thorough understanding of accurate classification.

Our detection model, which aims to reliably and precisely distinguish between phishing and authentic URLs, is trained using the extracted attributes as its basis. Making use of the wide range of parameters readily accessible our model applies sophisticated machine-learning techniques to identify minute details and irregularities indicative of phishing activities. Furthermore, by using deep learning techniques, our model is better equipped to detect malicious URLs since it can find intricate patterns and relationships in the data. Our research attempts to clear the path for more powerful and efficient defenses against the ubiquitous threat of phishing assaults in the digital realm by using this integrated and meticulously developed strategy. Figure 2 displays the model overview.

Feature extractor is designed to include the contextual sentiment score for each URL by using the GLOVE and Natural Language Tool Kit (NLTK) tools. Each URL in the dataset will get preprocessed and tokenized. The preprocessing includes the removal of stopping words, trimming, etc. Then each token will be passed on to the sci-kit learns text vectorizers to get a sentiment score. It is not possible to feed a sequence of symbols in raw data directly into a sci-kit feature extractor. Consequently, while some of them assume unstructured text documents of different lengths, most of them assume numerical feature vectors of a given size. Sci-kit-learn offers tools for the most popular methods of extracting numerical features from text to handle this, such as:

- **tokenizing**: The strings are tokenized, in which each potential token is assigned an integer id. The token separators may include whitespaces and punctuation marks.
- **counting**: The number of times each token appears in a document is recorded.
- **normalizing**: Involves weighting and normalising the tokens according to decreasing significance to those that appear in most samples.

Here are the definitions for features and samples: The frequency with which each unique token appears (normalised or not) is considered a feature. For a particular document, the vector containing all of those token frequencies is regarded as a multivariate sample.

B. FEATURE VECTORIZATION

Vectorization is the process of converting a set of text documents into numerical feature vectors. In this process, a matrix can be used to represent a corpus of documents, where each row represents a document and tokens are represented in each column. Tokenization, counting, and normalisation combined into "Bag of Words" or "Bag of n-grams" representation. Word occurrences are used to characterise the documents, with no consideration given to the terms' relative positions within the text.

Some terms will be prevalent in massive text corpus; hence, it has relatively little significant information about the document's real contents. Usually, one applies the TF-IDF transform to re-weight the count features into floating point values suitable for a classifier. Terms are denoted by the sign Tf, while inverted document frequencies are indicated by the notation Tf–idf.

For example, TF-IDF Transformer (norm ='l2', use_idf = True, smooth_idf = True, sublinear_tf = False) might be used with its default parameters. The term frequency is defined as the number of times a word appears in a particular document. It is multiplied by the idf component and is calculated as:

$$idf(x) = log\frac{1+n}{1+df(x)} + 1 \tag{1}$$

In the document set, df(x) is the number of documents that include word x, and n is the total number of documents in the document set. Subsequently, the Euclidean norm is used to normalize the resulting TF-IDF vectors.

$$u_{norm} = \frac{u}{||u||^2} = \frac{u}{\sqrt{u1^2 + u2^2 + \ldots.. + un^2}} \tag{2}$$

The term weighting method was initially created for information retrieval and is useful for grouping and classifying documents. The calculation of TF-IDF is shown in the next section.

$$idf(x) = log\frac{n}{1+df(x)} \tag{3}$$

In positional feature extraction the tfid transformer gets the weight by the token position on the glove dataset which will be defined by NLTK tool kit and assigned by the sci-kit transformers. There is a need for normalization because the matrix acquired during feature extraction contains floating point values. Min-max normalization operation rescales a set of data. The original set's smallest value would be mapped to 0. The largest value in the original set would be assigned the value 1. Every other value would be assigned a value between these two bounds. The lower bound is denoted by min(y) and the upper bound is denoted by max(y). The normalized value (y') can be represented as:

$$y' = \frac{y - min(y)}{max(y) - min(y)} \tag{4}$$

Followed by normalization, the entire dataset is divided into training and test sets of 80:20.

C. PSEUDOCODE

The goal of the proposed Phishing URL Detection Algorithm, presented in Algorithm 1, is to provide a dependable model for phishing URL detection. To guarantee data quality, the method begins with preprocessing a dataset D made up of URLs to eliminate null values and duplicates. It then utilizes a feature extraction technique to identify characteristics indicative of phishing activity from every URL in D. Afterwards, a list L containing the features is created. The software appends to L the features it computes for every URL u in D. The technique leverages the characteristics gathered and stored in L to train a machine learning model M after processing each URL. The output is then this trained model M, which can accurately identify URLs as phishing or authentic based on attributes that have been extracted. The algorithm provides a systematic framework for building a phishing detection model, leveraging machine learning techniques to enhance cybersecurity measures against fraudulent online activities.

Algorithm 1 Phishing URL Detection Algorithm

```
Require: URL dataset $D$
Ensure: Phishing detection model $M$
1: Preprocess dataset $D$ to remove duplicates and null
   values
2: Extract features from URLs in $D$ using feature
   extraction algorithm
3: Initialize empty list $L$
4: for each URL $u$ in $D$ do
5:     Calculate features for URL $u$
6:     Append features to $L$
7: end for
8: Train machine learning model $M$ using features in $L$
9: return Trained model $M$
```

IV. RESIDUAL PIPELINE

Residual pipeline enhances the overall system performance and is a crucial component of the entire architecture. To address the issue of the vanishing gradient, residual blocks were introduced. Skip connection is the technique primarily used here, it connects layer activations to subsequent layers by skipping portions of the intermediate layers. Regularisation will bypass any layer that reduces architecture performance, which is an advantage of using this type of skip link.

Figure 3 shows, the overview of the residual pipeline block. The input to the residual pipeline includes 27 URL properties, 64 filters, and two classes. It consists of convolutional blocks and seven inverted residual blocks which execute asynchronously. The convolutional block consists of a 3 × 3 convolution layer followed by a batch normalization layer, where the batch size chosen is 32. The ReLU activation function turns the provided input to the necessary output with the specified range. The output matrix from the convolutional block is fed onto the inverted residual blocks, which conduct different operations including convolution, separable convolution, batch normalization and activation.

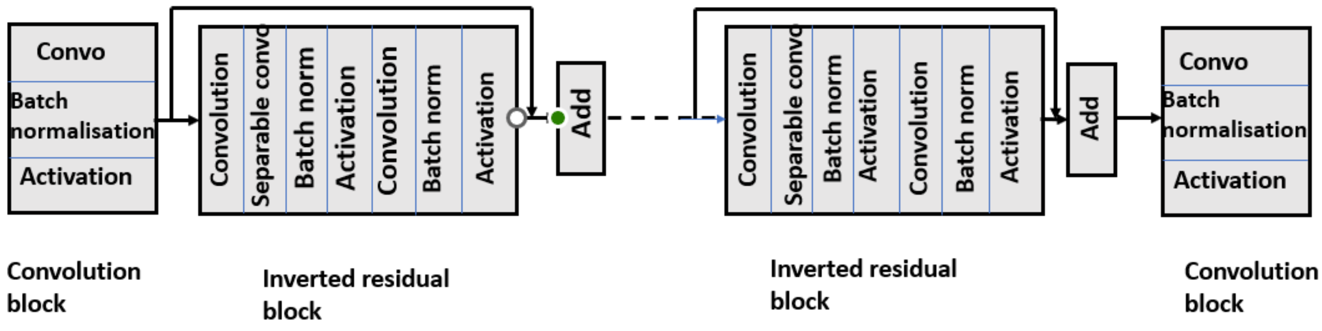


FIGURE 3. **Residual pipeline.**

Convolution operation performs computation in just a single step while separable convolution operates in two steps. Separable convolution divides the kernel into two smaller kernels, initially, the input is convoluted using the first kernel then the result obtained from convoluting the first kernel is again convoluted using the second kernel. It allows for the separation of even the smallest differences in data. Then the result after separable convolution is batch normalized and activated. After the process of activation, the result is again convoluted followed by batch normalization and activation. Finally, another convolutional block receives the output from each of these inverted residual blocks and the result is convoluted, batch-normalized, and activated. The output obtained after applying the residual pipeline will also be a matrix which will then be passed on to an MLP output layer.

V. OUTPUT BLOCK

After obtaining the result from the residual block, the output block comes into action. Figure 4 depicts the output block structure. Here, max pooling is performed initially and it determines the maximum value to gradually shrink the spatial size representation. Then the pooled matrix is flattened into a single column. Following the process of flattening, a neural network is used to process the massive input data vector for further purposes. The dense layer, which is highly connected to the layer before it, works to change the output's dimension. Typically, a dropout layer is used after a dense layer. Finally, an activation is performed, softmax activation function is used here because it always returns a value between 0 and 1. As a result, very small or negative values can be mapped to 0.0 and very large values can be represented as 1.0 when given as the weighted total of the input. The result from the output block will be a floating point value. A threshold of 0.5 is set, values below the threshold are placed in the lower class, which equals 0 and others are placed in a higher class, which equals 1. Class 0 represents the benign URLs and class 1 represents the phishing URLs. A sample output is represented in Table 3.

VI. EXPERIMENTAL RESULTS AND DISCUSSIONS

A. EXPERIMENTAL DATA

The underlying experimental data is taken from the Kaggle dataset [26]. URLs and their types are included in the

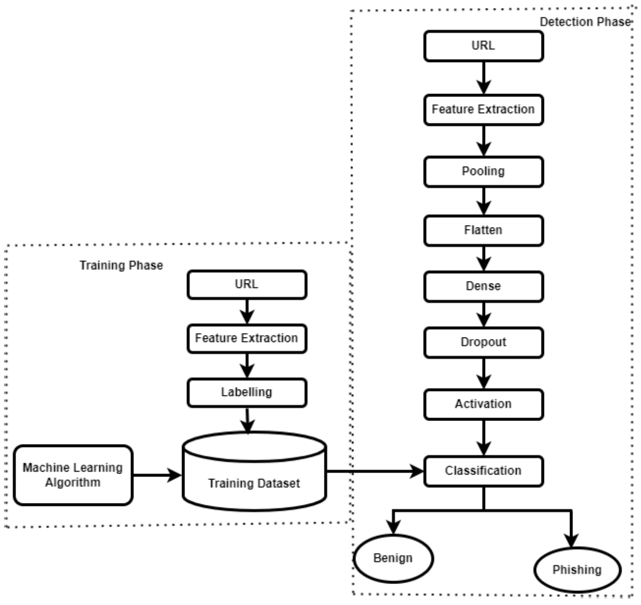


FIGURE 4. **Training and detection phases-output block.**

collection. Type indicates if a URL is phishing or safe. The dataset includes 6,51,191 URLs and their types; of these, 32,520 are malware URLs, 94,111 are phishing URLs, 96,457 are defacement URLs, and 4,28,103 are benign URLs. From this, only the benign and phishing URLs are selected for conducting the experiment, which constitutes 5,22,214 URL samples, among which 94,111 are phishing and 4,28,103 are benign. The sample dataset is shown in Table 4.

Batch size denotes the maximum number of URLs that our model can handle concurrently, while "epoch" denotes the number of training cycles that require the training set. Here, in this research study, the number of epochs is determined as 50. After the completion of each epoch, the loss is monitored, if the same error occurs for all fifty iterations then the execution gets stopped, which means the system is not correctly configured. Accuracy is monitored throughout the epochs and whenever best accuracy is observed then it is saved and the model is trained using this saved data.

TABLE 3. Result of output block.

No.	URL	Actual Type	Benign Prob	Phishing Prob	Predicted Type
0	zimbio.com/Marshon+Brooks	Benign	1	2.27077E-09	Benign
1	faqs.ign.com/objects/142/14260594.html	Benign	1	1.56892E-10	Benign
2	synaptixcommunications.com/index.php? %2FShowcase%2Fsimple-and-clear-educating-the- market%2FPrint.html	Phishing	5.56884E-12	1	Phishing
3	nephotography.com.au/galleries/index.php? do=registry	Phishing	9.25752E-17	1	Phishing
4	lifeskate.com/skate/2009/01/jessica-dub%C3%A9- and-bryce-davison-win-pairs-event-at-2009- canadian-figure-skating-championships.html	Benign	0.99920553	0.000794497	Benign
5	theonion.com/	Benign	5.96735E-09	1	Phishing
6	maxalbums.com/index.php?artist= Loredana%20Groza%20Zaraza	Benign	0.999999881	1.03087E-07	Benign
7	daylife.com/topic/Gabriel_Aubry	Benign	1	1.23556E-10	Benign
8	vale-healthcare.com/hand-clinic/	Benign	1	4.13954E-11	Benign
9	nudecelebsnow.com/tags/1/victoria+ justus.htm	Benign	1	7.10652E-11	Benign
10	0068555.com/cl/?module=System&method = LiveTop&args=livehall	Phishing	5.44753E-12	1	Phishing
11	local.yahoo.com/info-42862457-oakland- international-airport-oakland	Benign	0.998790324	0.001209757	Benign
12	areawidenews.com/story/1495635.html	Benign	1	1.8338E-10	Benign
13	czdesign.cz/kalenda/search.form/2013/01/ 31/-.html	Phishing	8.79956E-15	1	Phishing
14	jennieo.com/	Benign	1.64039E-08	1	Phishing
15	wn.com/American_Football_League	Benign	1	2.21074E-09	Benign
16	sbnation.com/ncaa-football/players/36729/brandon- smith	Benign	1	2.08677E-10	Benign
17	ababmx.com/index.php?page=default/hallof fame&year=1999	Benign	0.999999881	1.22945E-07	Benign
18	kalispellchristiancenter.org/team_bishop.html	Benign	0.999999881	1.32431E-07	Benign
19	tvguide.com/celebrities/peter-keleghan/169984	Benign	1	2.28043E-11	Benign
20	http://www.lescoulissesdetanger.com/galeriesmusees	Phishing	1	7.15947E-11	Phishing
21	perezhilton.com/category/william-shatner/	Benign	1	1.65901E-10	Benign
22	http://9779.info/%E5%B9%BC%E5%84%BF%E7 %B2 %BD%E5%8F%B6%E8%B4%B4%E7%94%BB/	Phishing	1	2.09898E-19	Phishing
23	http://babal.net/downloads_details/497/%D9%83%D8 %A7%D8%B8%D9%85- %D8%A7%D9%84%D8%B3%D8%A7%D9%87%D8 %B1—%D8%A7%D9%86%D8%AA- %D8%A7%D9%84%D8%AE%D8%A7%D8%B3 %D8%B1	Benign	0.978444874	0.021555193	Benign
24	hot-people.info/Crossett_AR_Dorothy-Moore_Eric– Newton_1051.html	Benign	1	0.993773699	Benign
25	ertx.com/movie/2008/mad-about-mambo	Benign	1	3.95767E-11	Benign

B. IMPLEMENTATION OF DOMAIN AGE ANALYSIS

Adding domain age analysis to our proposed approach is a critical step in improving the robustness and efficiency of our detection model. This feature is implemented by first gathering **WHOIS** information for every URL in our collection, and then determining the age of the domain that hosts the URL.

For each URL in our dataset, we use WHOIS data to obtain detailed information on domain registration. Gaining access to this data allows us to learn important things about the age and legitimacy of the URL-associated domains. Using the WHOIS information, we can find out the exact date that the domain for each URL was created. For purposes of determining the domain's age, this creation date is considered as the starting point. Next, we find out how old the domain is in days, months, or years by subtracting the current date from the date when the domain was created. The level of detail needed for our study dictates the interval between the two dates.

Adding domain age analysis to our detection process significantly enhanced and improved the detection accuracy of our model. Our model can differentiate between real and malicious URLs by looking at the age of the domain that hosts them. The analysis of domain age has had a significant impact on our results such as:

- Addition of domain age analysis has led to a big drop in false results. Knowing the difference between real websites and phishing URLs has made our model more accurate and lowered the number of false positives.
- Our method finds and avoids future computer threats by looking at domain ages. Our ability to find more things has improved. To help stop phishing attempts

TABLE 4. Sample dataset.

URL	Type
br-icloud.com.br	Phishing
mp3raid.com/music/krizz_kaliko.html	Benign
bopsecrets.org/rexroth/cr/1.htm	Benign
espn.go.com/nba/player/_/id/3457/brandon-rush	Benign
yourbittorrent.com/?q=anthony-hamilton-soulife	Benign
allmusic.com/album/crazy-from-the-heat-r16990	Benign
corporationwiki.com/Ohio/Columbus/frank-s-benson-P3333917.aspx	Benign
myspace.com/video/vid/30602581	Benign
quickfacts.census.gov/qfd/maps/iowa_map.html	Benign
nugget.ca/ArticleDisplay.aspx?archive=true&e=1160966	Benign
uk.linkedin.com/pub/steve-rubenstein/8/718/755	Benign
baseball-reference.com/players/h/harrige01.shtml	Benign
signin.eby.de.zukruygxctzmmqi.civpro.co.za	Phishing

TABLE 5. Comparison of phishing URL detection models.

Models	Performance Metrics (%)			
	Accuracy	Precision	Recall	F1 Score
Random Forest Classifier	85	78	82	80
Rule-Based Heuristics	80	72	75	73
Deep Neural Network	88	80	85	82
Gradient Boosting Classifier	82	75	78	76
Domain Age Analysis	95	88	91	89

from newly registered domains, our method improves identification by finding these domains.

We checked our model against four other models that were already made to find fake URLs to make sure our program worked. Our proposed model, which uses domain age analysis, was tested along with a number of other phishing URL detection methods. These models included random forests, gradient-boosting classifiers, neural networks, and rule-based strategies. The comparison used important performance measures like F1 Score, recall, accuracy, and precision. Table 5, which shows how our model is better than the current models.

Based on the comparison table, our model consistently offers more effectively than other methods at finding phishing URLs across all criteria. Our approach improves phishing attack security by improving recall, accuracy, precision, and F1 Score.

C. RESULT ANALYSIS & PERFORMANCE COMPARISON

We analyzed how well this model works using four different measures, including accuracy, recall, precision, and F1 Score [27]. The following metrics can be expressed using the following: True Positive (TrP) for the percentage of correctly identified phishing URLs, True Negative (TrN) for the percentage of legitimate URLs that are recognized as legitimate, False Positive (FaP) for the percentage of legitimate URLs that are mistakenly classified as phishing, and False Negative (FaN) for the percentage of legitimate URLs that are mistakenly identified as phishing.

$$Accuracy = \frac{TrP + TrN}{TrP + TrN + FaP + FaN} \tag{5}$$

$$Precision = \frac{TrP}{TrP + FaP} \tag{6}$$

$$Recall = \frac{TrP}{TrP + FaN} \tag{7}$$

$$F1\ \text{Score} = \frac{2 * Precision * Recall}{Precision + Recall} \tag{8}$$

The Accuracy measure in the metrics above represents the accurate classification proportion. Recall is the proportion of phishing URLs that we correctly identify out of all phishing URLs. The percentage of phishing URLs that we properly identify out of all the anticipated phishing URLs is known as precision. The harmonic average of the recall and precision rate is what determines the F1 Score.

A CNN [28], [30] and MHSA [31], [32] combined approach for detecting phishing websites has been selected as the baseline model to assess the efficacy of the proposed phishing URL detection method. After receiving URL strings as input data, CNN-MHSA [33], [34] passes them on to the embedding layer [35], [36], where one-hot encoding is carried out and the resulting matrix's dimension is subsequently reduced. The convolutional neural network receives the matrix after which it is fed for feature extraction [37]. The MHSA is then used to calculate the weight. On the training set, the baseline and suggested models are trained, and on the testing set, they are assessed. Figure 5 and Figure 6 shows a comparison of both models concerning the observed accuracy, precision, recall, and F1 Score and ROC AUC values. This shows that URL detection using residual pipelining has better performance in terms of all the chosen metrics. Then we compare the proposed model with different textual content features. The Table 6 presents the performance metrics of various classifiers on different types of textual content features for a classification task. Each classifier is evaluated based on precision, recall, F-score, area under the ROC curve (AUC), and accuracy. LR, XGBoost, Random Forest, Naive Bayes, DNN, LSTM are among the classifiers. Many textual content elements are taken into consideration, including count vectors, word sequence vectors, character sequence vectors, TF-IDF word level, TF-IDF N-gram level, and TF-IDF character level [38].

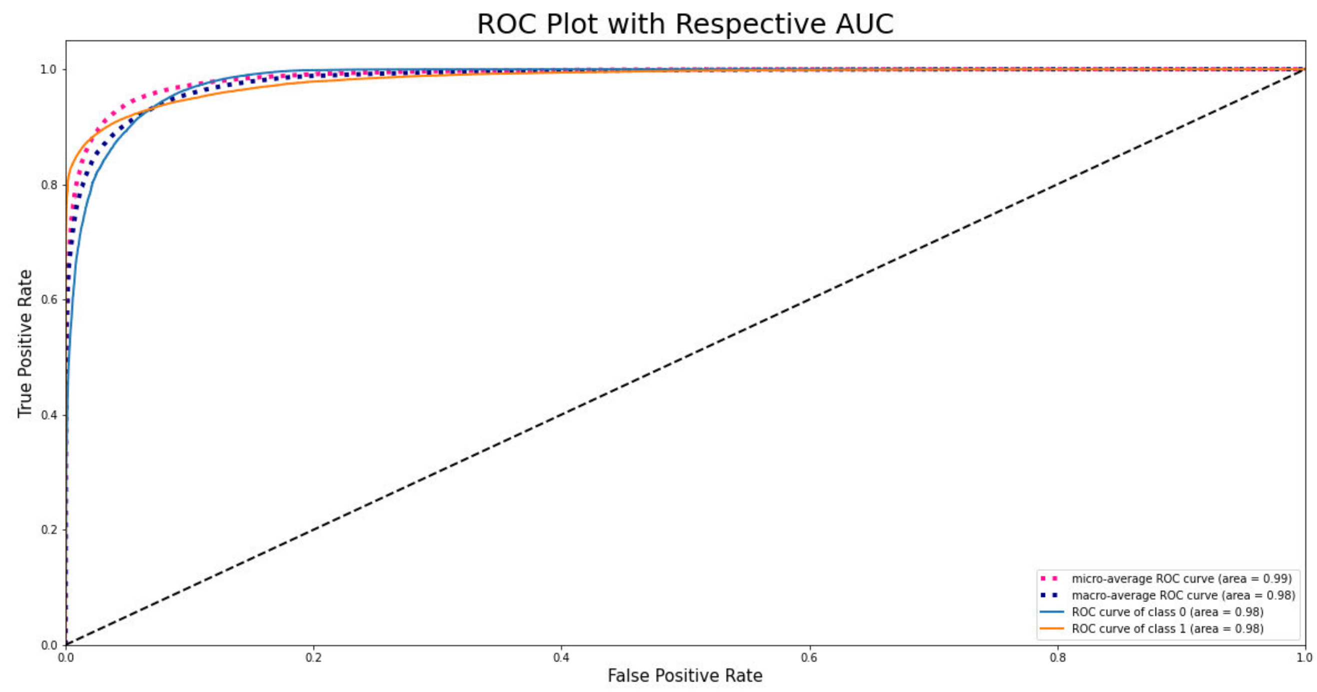


FIGURE 5. ROC curve.

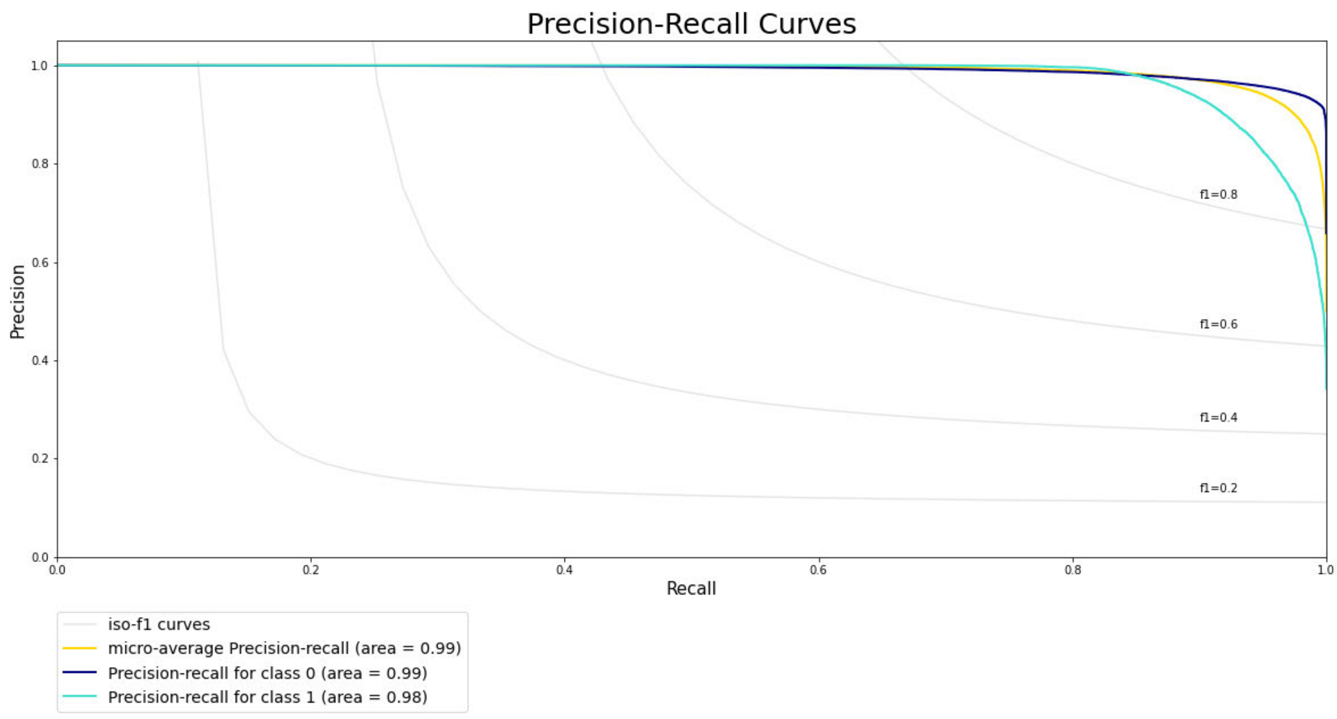


FIGURE 6. Precision-recall curve.

Different textual content features are representations of text data used as input for machine learning algorithms, each capturing distinct aspects of the text. TF-IDF at the word level assesses the significance of individual words in a document relative to a collection, while TF-IDF at the N-gram level considers sequences of words or characters. Character patterns are analysed using TF-IDF character level representation, which is helpful for languages with

TABLE 6. Classifier performance on textual content features.

Classifier	Textual Content Features(Tf-IDF)	Precision (%)	Recall (%)	F1 Score (%)	AUC (%)	Accuracy (%)
LR	word level	85.68	88.25	86.95	85.38	85.62
	N-gram level	85.23	85.42	85.33	83.93	84.05
	character level	84.55	87.15	85.83	84.13	84.39
	Count vectors	86.84	79.12	82.80	82.45	82.16
XGBoost	word level	88.44	88.56	88.50	87.41	87.52
	N-gram level	87.77	86.51	87.13	86.10	86.14
	character level	89.01	90.58	89.79	88.65	88.82
	Word sequences vectors	82.66	85.87	84.23	82.24	82.55
	Count vectors	88.26	87.75	88.00	86.95	87.02
Random Forest	word level	86.23	93.14	90.02	88.19	88.50
	N-gram level	87.11	90.28	88.68	87.02	87.21
	character level	85.76	93.45	89.82	87.83	88.01
	Count vectors	86.52	93.72	90.08	88.26	88.65
	Word sequences vectors	82.34	91.03	86.91	84.78	85.13
Naive Bayes	word level	85.20	79.91	82.42	81.08	81.15
	N-gram level	83.76	72.41	77.88	77.96	77.51
	character level	77.93	82.21	80.01	76.92	77.35
	Count vectors	84.28	73.22	78.31	78.49	78.05
	Word sequences vectors	65.12	46.28	54.12	59.68	58.74
Deep Neural Network	word level	88.37	92.51	90.39	88.86	89.02
	T N-gram level	89.14	85.86	87.46	86.81	86.65
	character level	89.52	92.01	90.74	89.28	89.43
	Count vectors	88.63	90.71	89.66	88.33	88.50
	Word sequences vectors	56.78	57.00	56.42	52.80	56.63
	Character sequences vectors	78.92	92.12	85.03	80.92	81.35
LSTM	word level	88.71	91.82	90.23	88.98	89.12
	N-gram level	89.32	90.10	89.71	88.54	88.71
	character level	83.52	90.14	86.48	84.39	84.73
	Count vectors	90.12	86.40	88.23	87.12	87.01
ResMLP	word level	98.10	96.20	98.14	98.75	98.90
	N-gram level	97.86	98.45	98.65	98.21	98.35
	character level	98.02	98.86	98.94	97.12	98.20
	Count vectors	97.74	98.91	98.32	98.05	98.15
	Word sequences vectors	96.65	98.12	97.38	97.02	96.20
	Character sequences vectors	97.32	98.25	98.76	97.98	98.05

complex morphology. By counting the instances of words in documents, count vectors offer efficiency and simplicity in situations where word frequency is crucial. Because word sequence vectors maintain word order when encoding word sequences, they are essential for applications like text generation [39]. Character sequence vectors are useful for analyzing complex writing systems and identifying misspelled words since character sequences are encoded to represent text. The best representation strategy must be found through experimentation because it depends on several variables, including properties, task complexity, and algorithm requirements.

From Table 6, it can be observed that the performance varies depending on the classifier and the type of textual content features used. For instance, MLP consistently achieves high precision, recall, F1 Score, AUC, and accuracy across different types of textual content features, indicating its robustness and effectiveness in capturing complex patterns in the data. When it comes to more complex textual content features, such as word sequence vectors and character sequence vectors, alternative classifiers do better than Naive Bayes. Text categorization tasks show how different classifiers work

TABLE 7. Comparison of phishing URL detection models.

Model	Accuracy	Precision	Recall	F1 Score
LR(Baseline)	0.85	0.78	0.82	0.80
LR(Enhanced)	0.88	0.82	0.86	0.84
XGBoost(Baseline)	0.83	0.75	0.80	0.77
XGBoost(Enhanced)	0.89	0.84	0.88	0.86
Random Forest (Baseline)	0.87	0.80	0.85	0.83
Random Forest (Enhanced)	0.91	0.86	0.90	0.88
Naive Bayes (Baseline)	0.78	0.70	0.75	0.72
Naive Bayes (Enhanced)	0.82	0.76	0.80	0.78
DNN (Baseline)	0.88	0.82	0.86	0.84
DNN (Enhanced)	0.92	0.87	0.90	0.88
LSTM (Baseline)	0.86	0.79	0.83	0.81
LSTM (Enhanced)	0.90	0.84	0.88	0.86
ResMLP (Baseline)	0.90	0.84	0.88	0.86
ResMLP (Enhanced)	0.98	0.93	0.96	0.94

by looking at the textual content parts. Figure 7 offers a visual representation of the comparisons for each group.

D. LEXICAL ANALYSIS OF URL STRUCTURE

Our proposed approach to find phishing URLs involves thoroughly studying the structure of the URL's words and looking for small connections that could mean a phishing attempt. From the URL's domain name, route, and query

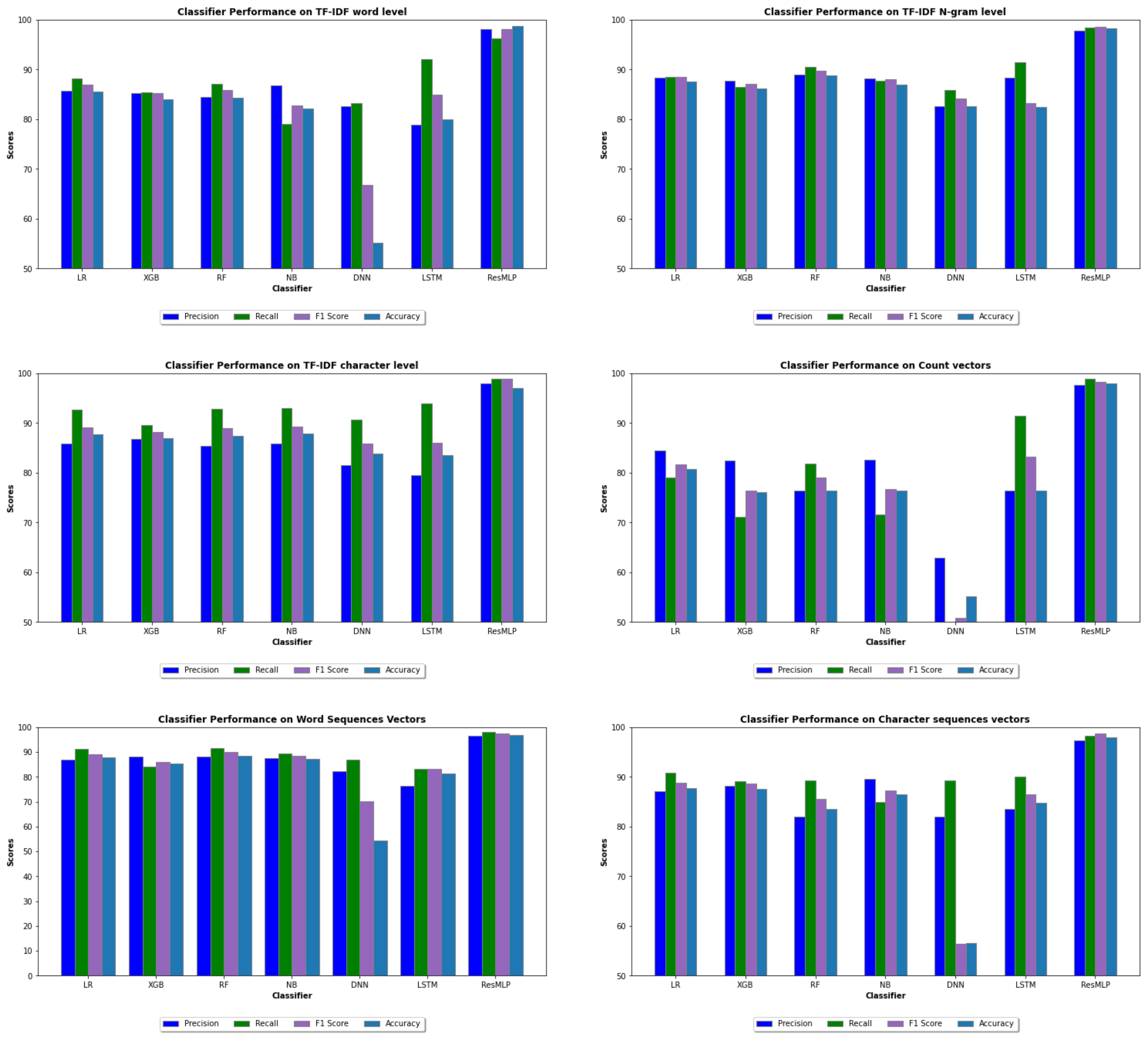


FIGURE 7. **Comparison of precision, recall, F1 score, and accuracy for various classifiers across different feature representations. Each subplot represents the performance metrics for a specific feature representation category, including TF-IDF word level, TF-IDF N-gram level, TF-IDF character level, count vectors, word sequences vectors, and character sequences vectors.**

parameters, we look for patterns in their syntax and meanings in this part.

Checking the domain name for any misspellings or ambiguities helps us spot scam attempts. Phishing websites often use attacks that are similar to those used by real domains. For example, attackers may change language or characters slightly. Our approach is designed to detect these unusual occurrences and identify potentially hazardous URLs [40].

The query parameters and URL route are the first places we search for signs of phishing attempts. Phishing URLs may use a convoluted path structure or a large number of query parameters to hide their true intent. Our model can analyze user input and detect abnormal activity, such potential phishing attempts.

Our analysis of URLs is comprehensive, covering both syntactic and semantic aspects. Just by examining the context and meaning of the URL sections, you may be able to find semantic inconsistencies or conflicts. Malicious URLs use domain names that don't relate to the content of the webpage or have an unusual combination of path segments and query parameters.

We incorporate lexical analysis of the URL structure into our detection method to enhance our model's understanding of URL properties and their security ramifications. This enhanced analysis allows our computer to detect phishing attempts despite the presence of minute signals that would be missed by previous detection techniques.

The method we propose can consistently differentiate between legitimate and malicious URLs by analyzing them component by component. Our approach is designed to identify suspicious patterns and highlight them, enhancing the accuracy of phishing attempt detection while lowering false positives and negatives. We enhanced the detection system's ability to handle newly emerging cyber threats by incorporating lexical analysis into our model. Our technology is designed to effortlessly handle even the most advanced phishing techniques, thanks to its continuous learning and discovery of new patterns that may indicate malicious activity. We can conduct experiments with and without this feature to evaluate performance and examine results using lexical analysis of URL structure. A comparison of performance is presented in Table 7.

VII. CONCLUSION AND FUTURE WORK

Phishing website assaults are a serious and growing risk to Internet users, as seen by the rise in incidents in recent times. Daily and hourly, a multitude of users inadvertently engage with phishing URLs, perpetuating the risk of cyber exploitation. Exploiters favor phishing as it exploits human vulnerabilities, exploiting the innate trust users place in seemingly authentic links, and evading conventional security measures. Although extensive research endeavors have been undertaken to counter these threats, achieving optimal detection accuracy remains an ongoing pursuit.

This research work aims to discern and categorize URLs into either phishing or benign classes. Evaluation metrics encompassing Accuracy, Precision, Recall, and F1 Score underscore the superior performance of the proposed system. Looking ahead, future endeavors may explore the expansion of this work into a multi-class classification framework. Meanwhile, efforts to optimize the residual pipeline, which currently comprises seven inverted residual blocks, will focus on streamlining and reducing the complexity of this architectural component.

REFERENCES

[1] A. Van der Merwe, M. Loock, and M. Dabrowski, "Characteristics and responsibilities involved in a phishing attack," in *Proc. Winter Int. Symp. Inf. Commun. Technol.*, 2005, pp. 249–254.

[2] (4th Quart., 2021). *APWG Phishing Activity Trends Report*. [Online]. Available: https://docs.apwg.org/reports/apwg/_trends_report_q4_2021.pdf

[3] B. Liang, M. Su, W. You, W. Shi, and G. Yang, "Cracking classifiers for evasion: A case study on the Google's phishing pages filter," in *Proc. Int. Conf. World Wide Web (WWW)*, Montral, QC, Canada, 2016, pp. 345–356.

[4] Q. Cui, G. V. Jourdan, G. V. Bochmann, R. Couturier, and I. V. Onut, "Tracking phishing attacks over time," in *Proc. 26th Int. Conf. World Wide Web (WWW)*, Perth, WA, Australia, 2017, pp. 667–676.

[5] H. Y. Abutair and A. Belghith, "Using case-based reasoning for phishing detection," *Proc. Comput. Sci.*, vol. 109, pp. 281–288, Jan. 2017.

[6] M. Al-Janabi, E. D. Quincey, and P. Andras, "Using supervised machine learning algorithms to detect suspicious URLs in online social networks," in *Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining*, Sydney, NSW, Australia, Jul. 2017, pp. 1104–1111.

[7] A. Blum, B. Wardman, T. Solorio, and G. Warner, "Lexical feature based phishing URL detection using online learning," in *Proc. 3rd ACM Workshop Artif. Intell. Secur.*, Chicago, IL, USA, Oct. 2010, pp. 54–60.

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

[9] R. Aravindhan, R. Shanmugalakshmi, K. Ramya, and C. Selvan, "Certain investigation on web application security: Phishing detection and phishing target discovery," in *Proc. 3rd Int. Conf. Adv. Comput. Commun. Syst. (ICACCS)*, Coimbatore, India, Jan. 2016, pp. 1–10.

[10] X. Xiao, D. Zhang, G. Hu, Y. Jiang, and S. Xia, "CNN–MHSA: A convolutional neural network and multi-head self-attention combined approach for detecting phishing websites," *Neural Netw.*, vol. 125, pp. 303–312, May 2020.

[11] W. Wang, F. Zhang, X. Luo, and S. Zhang, "PDRCNN: Precise phishing detection with recurrent convolutional neural networks," *Secur. Commun. Netw.*, vol. 2019, Oct. 2019, Art. no. 2595794, doi: 10.1155/2019/2595794.

[12] H. Zuhair and A. Selamat, "Phishing hybrid feature-based classifier by using recursive features subset selection and machine learning algorithms," in *Proc. 3rd Int. Conf. Reliable Inf. Commun. Technol. (IRICT)*. Springer, 2018, pp. 267–277.

[13] G. Ramesh, J. Gupta, and P. G. Gamya, "Identification of phishing webpages and its target domains by analyzing the feign relationship," *J. Inf. Secur. Appl.*, vol. 35, pp. 75–84, Aug. 2017, doi: 10.1016/j.jisa.2017.06.001.

[14] M. Cova, C. Kruegel, and G. Vigna, "There is no free phish: An analysis of 'free' and live phishing kits," in *Proc. WOOT*, Jul. 2008, pp. 1–8.

[15] L. Wenyin, G. Liu, B. Qiu, and X. Quan, "Antiphishing through phishing target discovery," *IEEE Internet Comput.*, vol. 16, no. 2, pp. 52–61, Mar. 2012.

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

[17] S. Marchal, J. François, R. State, and T. Engel, "PhishStorm: Detecting phishing with streaming analytics," *IEEE Trans. Netw. Service Manage.*, vol. 11, no. 4, pp. 458–471, Dec. 2014.

[18] R. M. Mohammad, F. Thabtah, and L. McCluskey, "Predicting phishing websites based on self-structuring neural network," *Neural Comput. Appl.*, vol. 25, no. 2, pp. 443–458, Aug. 2014.

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

[20] J. Zhang, and X. Li, "Phishing detection method based on borderline-smote deep belief network," in *Security, Privacy, and Anonymity in Computation, Communication, and Storage (SpaCCS)* (Lecture Notes in Computer Science), vol. 10658, G. Wang, M. Atiquzzaman, Z. Yan, and K. K. Choo Eds. Cham, Switzerland: Springer, 2017, pp. 45–53.

[21] R. Verma and A. Das, "What's in a URL: Fast feature extraction and malicious URL detection," in *Proc. 3rd ACM Int. Workshop Secur. Privacy Anal.*, Scottsdale, Arizona, USA, Mar. 2017, pp. 55–63.

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

[23] H. Sun, Z. Liu, S. Wang, and H. Wang, "Adaptive attention-based graph representation learning to detect phishing accounts on the Ethereum blockchain," *IEEE Trans. Netw. Sci. Eng.*, vol. 11, no. 3, pp. 2963–2975, May 2024, doi: 10.1109/tnse.2024.3355089.

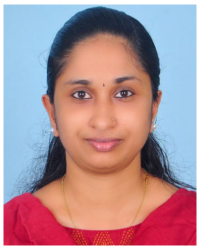
[24] M. W. Shaukat, R. Amin, M. M. A. Muslam, A. H. Alshehri, and J. Xie, "A hybrid approach for alluring ads phishing attack detection using machine learning," *Sensors*, vol. 23, no. 19, p. 8070, Sep. 2023.

[25] S. Asiri, Y. Xiao, S. Alzahrani, S. Li, and T. Li, "A survey of intelligent detection designs of HTML URL phishing attacks," *IEEE Access*, vol. 11, pp. 6421–6443, 2023.

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

[27] 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.

[28] X. Xiao, Z. Wang, Q. Li, S. Xia, and Y. Jiang, "Back-propagation neural network on Markov chains from system call sequences: A new approach for detecting Android malware with system call sequences," *IET Inf. Secur.*, vol. 11, no. 1, pp. 8–15, Jan. 2017.
[29] D. Antoniades, I. Polakis, G. Kontaxis, E. Athanasopoulos, S. Ioannidis, E. P. Markatos, and T. Karagiannis, "We.B: The web of short URLs," in *Proc. 20th Int. Conf. World Wide Web*, Mar. 2011, pp. 715–724.
[30] N. Ketkar and J. Moolayil, "Convolutional neural networks," in *Deep Learning with Python: Learn Best Practices of Deep Learning Models With PyTorch*, 2021, pp. 197–242.
[31] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Providence, RI, USA, Jun. 2012, pp. 3642–3649.
[32] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," Sep. 2014, *arXiv:1409.0473*.
[33] V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu, "Recurrent models of visual attention," in *Proc. 27th Int. Conf. Neural Inf. Process. Syst. (NIPS)*, Montreal, QC, Canada, 2014, pp. 2204–2212.
[34] M. T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," Aug. 2015, *arXiv:1508.04025*.
[35] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. 31st Conf. Neural Inf. Process. Syst.*, Long Beach, CA, USA, 2017, pp. 1–11.
[36] T. Berners-Lee, L. Masinter, and M. McCahill, *Uniform Resource Locators (URL)*, document RFC 106107, 1994.
[37] Y. Kim, "Convolutional neural networks for sentence classification," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, Doha, Qatar, 2014, pp. 1746–1751.
[38] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 770–778.
[39] M. J. Pillai, S. Remya, V. Devika, S. Ramasubbareddy, and Y. Cho, "Evasion attacks and defense mechanisms for machine learning-based web phishing classifiers," *IEEE Access*, vol. 12, pp. 19375–19387, 2024.
[40] S. Remya, M. J. Pillai, C. Arjun, S. Ramasubbareddy, and Y. Cho, "Enhancing security in LLNs using a hybrid trust-based intrusion detection system for RPL," *IEEE Access*, vol. 12, pp. 58836–58850, 2024, doi: 10.1109/access.2024.3391918.



S. REMYA received the Ph.D. degree in computer science and engineering from Vellore Institute of Technology, Vellore Campus. She is currently an Assistant Professor with the Department of Computer Science and Engineering, School of Computing, Amrita Vishwa Vidyapeetham, Amritapuri Campus, Kollam, Kerala, India. Her research interests include deep learning, data science, computer vision, security, and smart environments.

MANU J. PILLAI received the Ph.D. degree in computer science and engineering from the National Institute of Technology, Calicut. He is currently an Associate Professor with the Department of Computer Science and Engineering, TKM College of Engineering, Kollam, Kerala, India. His research interests include wireless networks, deep learning, and smart environments.

KAJAL K. NAIR received the B.Tech. degree from Kerala Technological University (KTU) and the M.Tech. degree from the TKM College of Engineering, Kollam, Kerala, where she demonstrated outstanding academic performance. Her research interest includes cybersecurity, with a specific focus on identifying phishing attacks.

SOMULA RAMA SUBBAREDDY received the Ph.D. degree in computer science and engineering from VIT University, Vellore, India, in 2022. He was a Postdoctoral Research with the Department of Information and Communication, Sunchon National University, South Korea, in 2024. He is currently an Assistant Professor with the Department of Information Technology, Vallurupalli Nageswara Rao Vignana Jyothi Institute of Engineering and Technology, Hyderabad. He has more than 40 publications in reputed journals and conferences. His research interests include mobile cloud computing, the IoT, machine learning, and edge computing.

YONG YUN CHO received the Ph.D. degree in computer engineering from Soongsil University. He is currently a Professor with the Department of Information and Communication Engineering, Sunchon National University. His main research interests include system software, embedded software, and ubiquitous computing.