PHISHING URL DETECTION USING RESIDUAL PIPELINING

PROJECT REPORT

Submitted by

KAJAL K NAIR

REG NO : TKM20CSCE08

In partial fulfillment for the award of the degree of

MASTER OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

Under the guidance of
Prof. Manu J Pillai

Thangal Kunju Musaliar College of Engineering
Kerala

SEPTEMBER 2022

Thangal Kunju Musaliar College of Engineering
Dept. of Computer Science & Engineering

C E R T I F I C A T E

This is to certify that this report titled ***Phishing URL detection using residual pipelining*** is a bonafide record of the **Project** presented by **KAJAL K NAIR (TKM20CSCE08)**, under our guidance and supervision, in partial fulfillment of the requirements for the award of the degree, **M.Tech in Computer Science & Engineering** in **APJ Abdul Kalam Technological University** .

Coordinator	Supervisor	Head of the Department
Dr. Ansamma John	Prof. Manu J Pillai	Dr. Dimple A Shajahan
Professor	Associate Professor	Associate Professor
Dept. of CSE	Dept. of CSE	Dept. of CSE
TKMCE	TKMCE	TKMCE

ACKNOWLEDGEMENT

A successful project is a fruitful culmination of efforts by many people, some directly involved and some others indirectly, by providing support and encouragement. Firstly I would like to thank the almighty for giving me the wisdom and grace for making my project a memorable one. I thank him for steering me to the shore of fulfillment under his protective wings.

I express my sincere gratitude to **Dr. T A Shahul Hameed**, Principal of T.K.M College of Engineering for giving me an opportunity to present my project. I would like to thank **Dr. Dimple A Shajahan**, Associate Professor and Head of the Department, CSE, TKMCE, for her constant support and encouragement throughout the work.

With a profound sense of gratitude, I would like to express my heartfelt thanks to my guide **Prof. Manu J Pillai**, Associate Professor, CSE, TKMCE, and project coordinator **Dr. Ansamma John**, Professor, CSE, TKMCE for their expert guidance, cooperation and immense encouragement. I also extend my thanks to the entire faculty members and staffs of the Department of Computer Science & Engineering, TKMCE, who has encouraged me throughout this work.

I also express my thanks to my loving parents and friends, for their support and encouragement in the successful completion of this project work.

KAJAL K NAIR

Abstract

Phishing websites are the ones with same name and appearance as an official website. Nowadays, URL phishing is increasing and are causing great threats, they pretend to be the same as the official website and steal user information. Earlier blacklists were used to identify these URLs; however, they cannot identify the one-time Uniform Resource Locators (URL). Deep learning methods are employed to avoid such risks and also to improve the detection accuracy and decrease the misjudgment ratio. In URL detection using residual pipelining, the common URL features along with some sentimental values are extracted and fed onto a residual pipeline. The residual pipeline consists of convolutional blocks and inverted residual blocks. The result obtained from this block is finally passed onto the MLP output layer where the actual detection and classification of an URL takes place. The dataset is obtained from Kaggle. The accuracy, precision, F1 score, and recall are being measured and it is observed that it is much higher than those of several other traditional algorithms

Contents

List of Figures

List of Tables

Chapter 1

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. These links typically take users to fraudulent, malware-infected websites that are looking for personal information, including passwords and banking information, which may result in significant losses for the victims[17]. Realistically, detecting and recognising phishing websites is a dynamic and difficult task. Phishing attacks can be conducted in a variety of ways, including via email, websites, spyware, SMS, and voice calls. One of the most common 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 has increased more than twice as much as it did in 2022. Anti-Phishing Working Group (APWG)[1] reported 316,747 attacks in December 2021, the highest monthly total in its history. According to APWG founding member OpSec Security, bank-related phishing attacks accounted for 23.2% of all phishing attacks in the fourth quarter of 2021.

Anti-phishing is the technique of preventing phishing attacks in which attackers try to get sensitive data by pretending to be a reliable source. 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[18] to reduce these hazards, however, these tools are not the last layer of defense. Anti-phishing software is a platform or collection of software services that recognises malicious inbound messages pretending to be from a reputable source

or attempting to gain trust through social engineering, enables corrective actions, and gives users the ability to build blacklists[24] and whitelists for message filtering. However, these are insufficient inorder to battle phishing, since attacker make use of one-time phishing URLs and direct users to visit before redirecting them to legitimate sites. Machine learning techniques, such as Support Vector Machines (SVM) and Random Forest (RF), are used to deal with this trick, relying on a built-in classifier to examine the properties of sample URLs[19][20] in order to make judgments for new, developing ones[21]. Likewise, deep learning based methods[25] are developed which can achieve better classification than that of conventional machine learning methods.

Motivated by this idea, residual pipelining-based phishing URL detection was developed, 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. Ultimately, rigorous testing reveals that the accuracy reaches up to 98.295%, which outperforms the conventional methods.

The remainder of this report is organized as follows. Chapter 2 recalls some related works used as references for completing this study. Chapter 3 includes all background information used to carry out this study. Chapter 4 presents the dataset used for conducting the experiment and results of the study. Lastly, Chapter 5 concludes and discusses future work.

Chapter 2

Related Works

Phishing has long been one of the most popular cyber-attack strategies used by bad actors. Phishing is an effort made by a person or a group of people to obtain personal information, including passwords, banking information, and credit card numbers. These problem posed by phishing websites has been addressed by numerous studies. Many techniques for detecting phishing websites have been proposed, including blacklist-based techniques and heuristic-based techniques. The statistics from the training dataset have a substantial impact on the weights of the heuristic in the heuristic-based approach. Blacklists[2], which is actually 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, graph-based, machine-learning-based, and deep learning-based techniques have been developed.

Xi Xiao et.al.[3] proposed a model named CNN-MHSA: A Convolutional Neural Network and multi-head self-attention combined approach for detecting phishing websites. Here the feature extraction and weight calculation are done separately by duplicating the input matrix into two, then with the aid of self attention mechanism websites are identified as malicious or benign.

Weiping Wang et.al.[4] proposed the model PDRCNN: Precise Phishing Detection with Recurrent Convolutional Neural Networks. PDRCNN encodes an URL information into a two-dimensional tensor and feeds the tensor into a deep learning neural network inorder to classify the original URL.

Zuhair, H. et.al.[5] constructed Phishing Hybrid Feature-Based Classifier (PHFBC) by Using Recursive Features Subset Selection and Machine Learning Algorithms. They extracted features manually from the phishing and legitimate websites. PHFBC which hybridized Naive Bayes and Decision Tree with a statistical criterion of Phish Ratio and a Recursive Feature Subset Selection Algorithm was also proposed to characterize phishing holistically

with a robust selected subset of features.

Ramesh et.al.[6] introduced a method named Identification of phishing webpages and its target domains by analyzing the feign relationship. They constructed a parasitic matrix inorder to show the relation between two sites and also used row sum and column sum to find target link of sites.

Cova et.al.[7] conducted an analysis to comprehend the basic layout and use of phishing kits, to recognise the obfuscation strategies fraudsters use to conceal their backdoors that they have planted and to locate and alert interested parties of the methods used by phishers to transmit phished data so that suitable defences can be put in place.

Liu, W. et.al.[8] introduced a method named Antiphishing through Phishing Target Discovery. It gathers websites that are either directly or indirectly linked to a specific questionable website. Thus making it possible to identify a webpage's so-called "parasitic" community and, eventually, its phishing target, or the webpage that has the strongest parasitic connection to the suspect webpage. Users can determine whether a given webpage is a phishing page by locating this target.

Zhang, Y. et.al.[9] designed a model called CANTINA: A Content-Based Approach to Detecting Phishing Web Sites. To determine character scores for each character and extract keywords from texts, TF-IDF counts the characters in each document as well as the total number of documents. To perform Google searches, the top 5 keywords from each website are chosen. The website is regarded as legitimate if it appears in the top results and shares the same domain.

Marchal et.al.[10] extracted 212 features from URL and HTML, and Gradient Boosting was chosen to construct the classification model. These techniques employ machine learning algorithms to train a model to determine if a URL is phishing or authentic by manually extracting characteristics from legitimate and phishing websites.

Mohammad et.al.[11] proposed a work named Predicting phishing websites based on self-structuring neural network. An artificial neural network-based intelligent model for predicting phishing attacks, particularly self-structuring neural networks was developed with a high tolerance for noisy input, high fault tolerance, and high forecast accuracy while automating the network construction process.

Nguyen et.al.[12] introduced a method named An efficient approach for

phishing detection using single-layer neural network. The value of heuristics is calculated and the weights are generated using a single-layer neural network.

Zhang and Li[13] proposed Phishing Detection Method Based on Borderline-Smote Deep Belief Network. The history of phishing prevention and control is analysed, and a Borderline-Smote (Synthetic Minority Over-sampling Technique) DBN (Deep Belief Network) technique to identify phishing is presented. The probability distribution is calculated through the edge distribution of energy function and maximum likelihood is estimated.

Verma and Das[14] introduced online learning with n-gram for the detection of phishing websites. The URLs are splited into grams and various online learning algorithms are used for the detection of phishing websites.

Yang et.al[15] developed a method named Phishing Website Detection Based on Multidimensional Features Driven by Deep Learning. Deep learning extracts character sequence features from a given URL and uses them for quick classification. URL is transformed into a matrix through One-hot encoding and the dimension of the matrix is reduced using embedding.

Chapter 3

Methodology

In the detection of phishing URLs, residual pipelining is introduced so that their strengths can be used to better support phishing website detection.

3.1 Feature Extraction

The feature extractor plays a vital role in the extraction of features from URLs. The input to feature extractor is a collection of URLs and their type. The type here indicates whether the URL is phishing or benign one. Initially, the dataset is preprocessed to remove duplicates and none values. The criteria used to identify whether a URL is dangerous or not are features taken from the URL. Here, URL properties[16] along with sentimental properties are choosen as features. 25 different properties of URL are selected which includes

- URL length - The number of characters used on the URL determines how long it is.
- Host Length - The URL's domain name's length
- Directory length - The length of folder or directory.
- 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 and https protocol.
- 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.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 - 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

- 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.

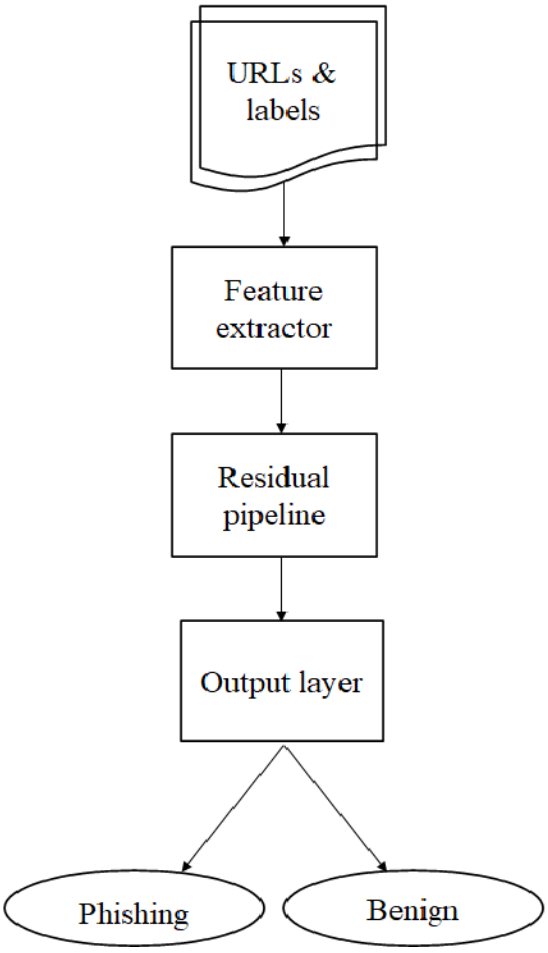

Figure 3.1: Model overview.

Feature extractor is designed to include the contextual sentiment score for each URL by using the GLOVE and Natural Language Tool Kit (NLTK) tools. NLTK is the most popular platform for creating Python programs that use human language data. Each URL in the dataset will get preprocessed and tokenized. The preprocess include 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. In a scikit feature extractor the raw data in a sequence of symbols cannot be fed directly onto the algorithms themselves. Since most of them expects numerical feature vectors with some fixed size rather than those raw text documents with variable length. In order to address this, utilities are provided by scikit-learn for the most common ways to extract numerical features from text content, namely:

- **tokenizing** :- the strings are tokenized and given an integer id for each possible token, for instance by using white-spaces and punctuation marks as token separators.
- **counting** :- occurrences of the tokens in each document are counted.
- **normalizing** :- normalizing and weighting the tokens with diminishing importance to those tokens that occur in the majority of samples.

Features and samples are defined as follows:

- frequency of occurrence of each individual tokens (normalized or not) is treated as a feature.
- the vector of all those token frequencies for a given document is treated as a multivariate sample.

By using a matrix ,a corpus of documents can be represented, with one row per document and one column per token occurring in the corpus. The general process of turning a collection of text documents into numerical feature vectors is termed as vectorization. This particular strategy i.e; tokenization, counting and normalization, is called the Bag of Words or "Bag of n-grams" representation. Documents are described by word occurrences while completely ignoring the relative position information of the words in the document.

In this approach large text corpus, some words will be very present, hence it carries very little meaningful information about the actual contents of the document. If the direct count data were feed directly to a classifier the very frequent terms shadows the frequencies of rarer yet more interesting terms. Mostly, tf-idf transform is used in-order to re-weight the count features into floating point values suitable for usage by a classifier. Tf means the term-frequency while tf–idf means the term-frequency times inverse document-frequency.

Using the Tfidf Transformer's default settings, Tfidf Transformer (norm='l2', use_idf = True, smooth_idf = True, sublinear_tf = False) the term frequency, the number of times a term occurs in a given document, is multiplied along with idf component, and is computed as

$$idf(t) = log\frac{1+n}{1+df(t)} + 1 \tag{3.1}$$

where n is the total number of documents in the document set, and df(t) is the number of documents in the document set that contain term t. The resulting tf-idf vectors are then normalized by the Euclidean norm:

$$v_{norm} = \frac{v}{||v||2} = \frac{v}{\sqrt{v1^2 + v2^2 + ... + vn^2}} \tag{3.2}$$

It was originally a term weighting scheme developed for information retrieval that has a good use in the document classification and clustering. The following part shows the computation of tf-idfs and the tf-idfs in scikit-learn's Tfidf Transformer and Tfidf Vectorizer differ slightly from the standard textbook notation that defines the idf as

$$idf(t) = log\frac{n}{1+df(t)} \tag{3.3}$$

In positional feature extraction the tfid transformer get the weight by the tokens position on the glove dataset which will be defined by NLTK tool kit and assigned by the scikit 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(x) and upper bound is denoted by max(x). The normalized value (x') can be represented as:

$$x' = \frac{x - min(x)}{max(x) - min(x)} \tag{3.4}$$

Followed by normalization, the entire dataset is divided into test set and train test. The training set is used to train the model, while the testing set is used to evaluate the model's accuracy, 80:20 split ratio is the most common. In other words, 80% of the dataset is used as the training set, and 20% is used as the testing set.

3.2 Residual Pipeline

Residual pipeline plays a very important role in the complete architecture, it improves the performance of overall system. The vanishing gradient is a prevalent issue in deep learning. This results in the gradient becoming zero or being overly large. As a result, as the number of layers rises, the training and test error rates also rise. 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. The benefit of including this kind of skip link is that regularisation will skip any layer that degrades architecture performance.

Fig.3.2 shows, the overview of residual pipeline block. The input to residual pipeline includes 27 URL properties, 64 filters and two classes. It consists of convolutional blocks and seven inverted residual blocks which executes asynchronously. The convolutional block consists of a 3 x 3 convolution layer followed by a batch normalisation layer, where the batch size choosen is 32 and an activation function. Activation function turns the provided input to the necessary output with the specified range. ReLU is the activation function used here since it outputs the input directly if it is a positive value else it will output zero. The output matrix from convolutional block is fed onto the inverted residual blocks, that conduct different operations including convolution, separable convolution, batch normalization and activation. Convolution operation performs computation in just a single step while separable convolution performs operation in two steps. Separable convolution divides the kernel into two smaller kernels, initially the input is convoluted using 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 residual pipeline will also be a matrix which will then be passed on to an MLP output layer..

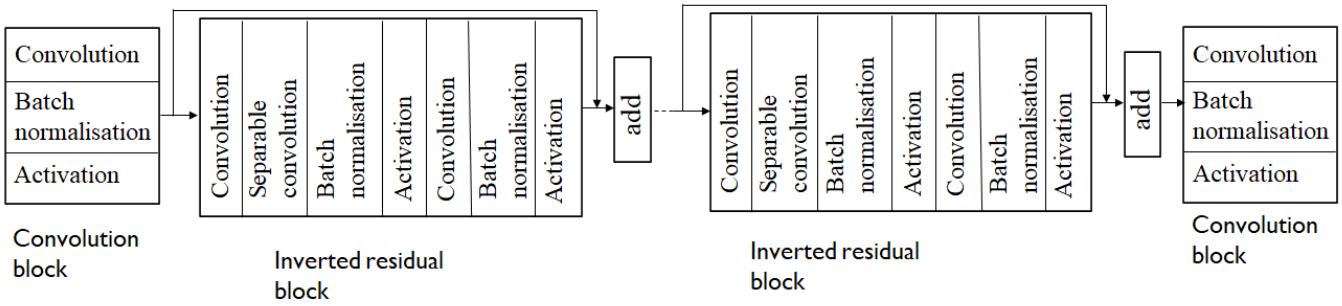

Figure 3.2: Residual Pipeline.

3.3 Output Block

After obtaining the result from residual block, output block comes into action. Fig.3.3 shows the structure of output block. At first, max pooling is performed. It determines the maximum value, the main purpose of this is to gradually shrink the representation's spatial size. Then the pooled matrix is flattened into a single column. After the flattening step, the long vector of input data is passed through a neural network for further processing. 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 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 higher class, which equals 1. Class 0 represents the benign URLs and class 1 represents the phishing URLs. A sample output is represented in Fig.3.4

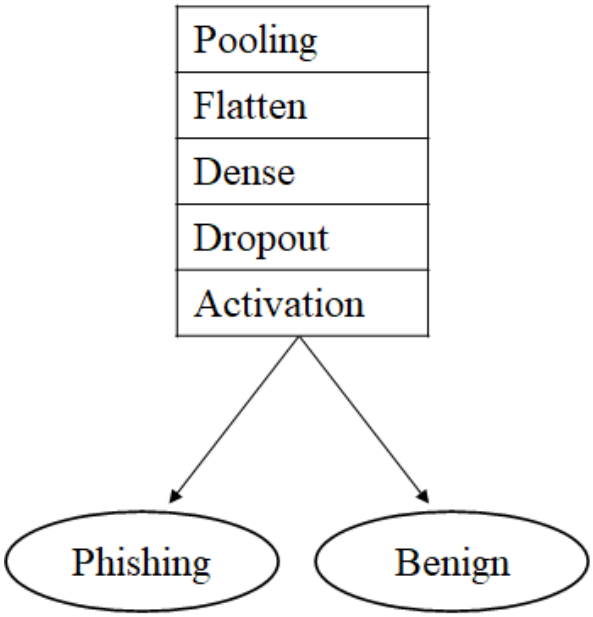

Figure 3.3: Output Block.

	A	B	C	D	E	F
		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		http://synaptixcommunications.com/index.php?%2FShowcase%2Fsimple-and-c	Phishing	5.56884E-12	1	Phishing
3		http://www.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-	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		http://www.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		http://www.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/halloffame&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	7.15947E-11	1	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	Phishing	2.09898E-19	1	Phishing
23		http://babal.net/downloads_details/497/%D9%83%D8%A7%D8%B8%D9%85-%D	Benign	0.978444874	0.021555193	Benign
24		hot-people.info/Crossett_AR_Dorothy-Moore_Eric--Newton_1051.html	Benign	0.993773699	0.006226318	Benign
25		ertx.com/movie/2008/mad-about-mambo	Benign	1	3.95767E-11	Benign

Figure 3.4: Result of output block.

Chapter 4

Experimental Results and Discussions

4.1 Experimental Data

The dataset obtained from Kaggle[22] is used as the base experimental data. The dataset contains URLs along with their type. Type represents whether the URL is benign or phishing. Dataset consists of 6,51,191 URLs and their type, out of which 4,28,103 are benign URLs, 96,457 are defacement URLs, 94,111 are phishing URLs, and 32,520 are malware URLs. From this only the benign and phishing URLs are selected for conducting the experiment, which constitutes of 5,22,214 URL samples, among which 94,111 are phishing and 4,28,103 are benign. The sample dataset is shown in Fig. 4.1.

Meanwhile, "batch size" refers to the number of URLs that our model can process at once, whereas "epoch" refers to the training rounds with the training data. Here, 50 is the number of epochs used. After completion of each epoch, the loss is monitored, if same error occurs for all the fifty iterations then the execution get stopped, which means the system is not correctly configured. Accuracy is monitored throughout the epochs and whenever best accuracy are observed then it is saved and the model is trained using this saved data.

4.2 Metrics

To evaluate the model 4 different metrices are used: Accuracy, Recall, Precision, and F-1[23] score, in order to completely describe the performance of the model. First, if we use True Positive (TP) to represent the percentage of phishing URLs that are correctly identified, True Negative (TN) to represent the percentage of legitimate URLs that are recognised as legitimate, False Positive (FP) to represent the percentage of legitimate URLs that are incorrectly classified as phishing, and False Negative (FN) to represent the percentage of phishing URLs that are identified as legitimate, then the met-

	url	type
0	br-icloud.com.br	phishing
1	mp3raid.com/music/krizz_kaliko.html	benign
2	bopsecrets.org/rexroth/cr/1.htm	benign
3	http://buzzfil.net/m/show-art/ils-etaient-loin...	benign
4	espn.go.com/nba/player/_/id/3457/brandon-rush	benign
5	yourbittorrent.com/?q=anthony-hamilton-soulife	benign
6	allmusic.com/album/crazy-from-the-heat-r16990	benign
7	corporationwiki.com/Ohio/Columbus/frank-s-bens...	benign
8	myspace.com/video/vid/30602581	benign
9	quickfacts.census.gov/qfd/maps/iowa_map.html	benign

Figure 4.1: Sample dataset

rices can be expressed as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

$$Precision = \frac{TP}{TP + FP} \tag{4.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.3}$$

$$F1score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{4.4}$$

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.

4.3 Performance Comparison

To evaluate the effectiveness of the proposed phishing URL detection method, a Convolutional Neural Network[26][27] and multi-head self-attention[28][29] combined approach for detecting phishing websites[3] is selected as the baseline model. In, CNN-MHSA[30][31], it first takes URL strings as the input data and is then passed onto the embedding layer[32][33] where one-hot encoding is performed and then the dimension of obtained matrix is reduced. The matrix is then fed onto convolutional neural network for feature extraction[34] and then weight calculation is done with the help of MHSA. The baseline and proposed models are trained on the training set and evaluated on the testing set. Figure 4.2. shows a comparison of both models with respect to the observed accuracy, precision, recall and F1-score. This shows that,the URL detection using residual pipelining has better performance in terms of all the four chosen metrics. Fig. 4.3 shows the ROC curve and Fig.4.4 shows the precision-recall curve. %hline

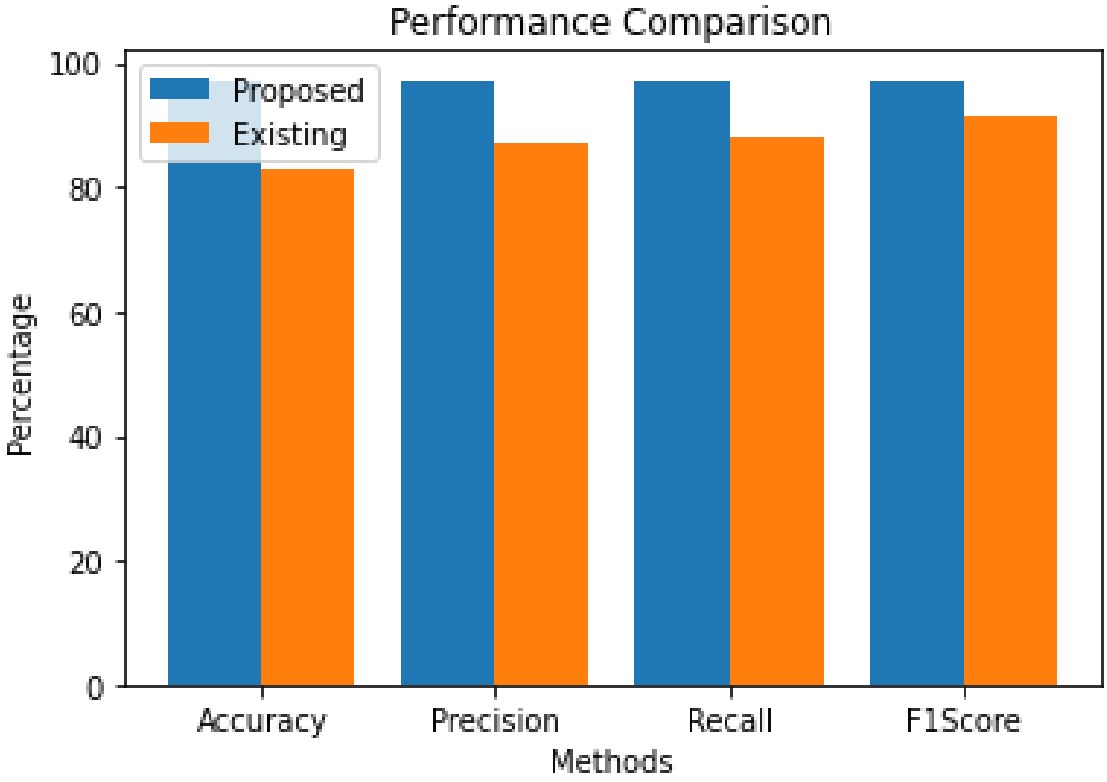

Figure 4.2: Performance Comparison

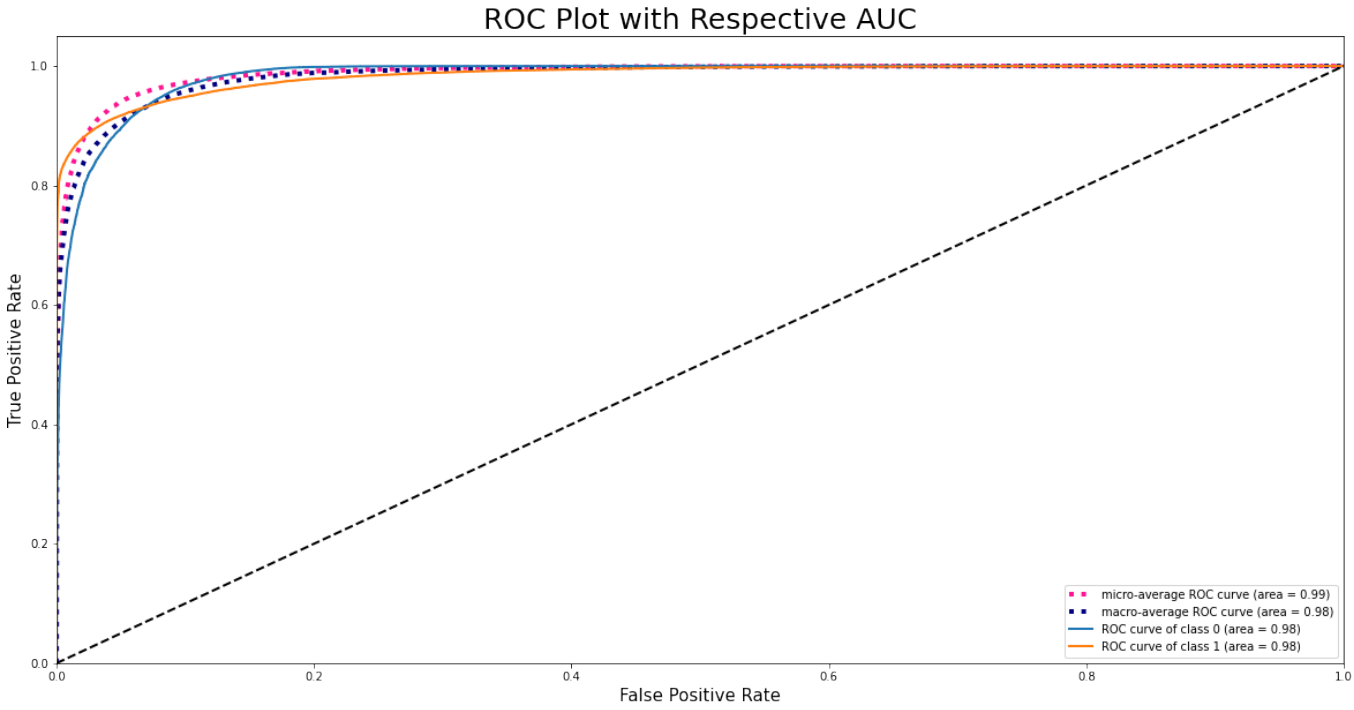

Figure 4.3: ROC curve

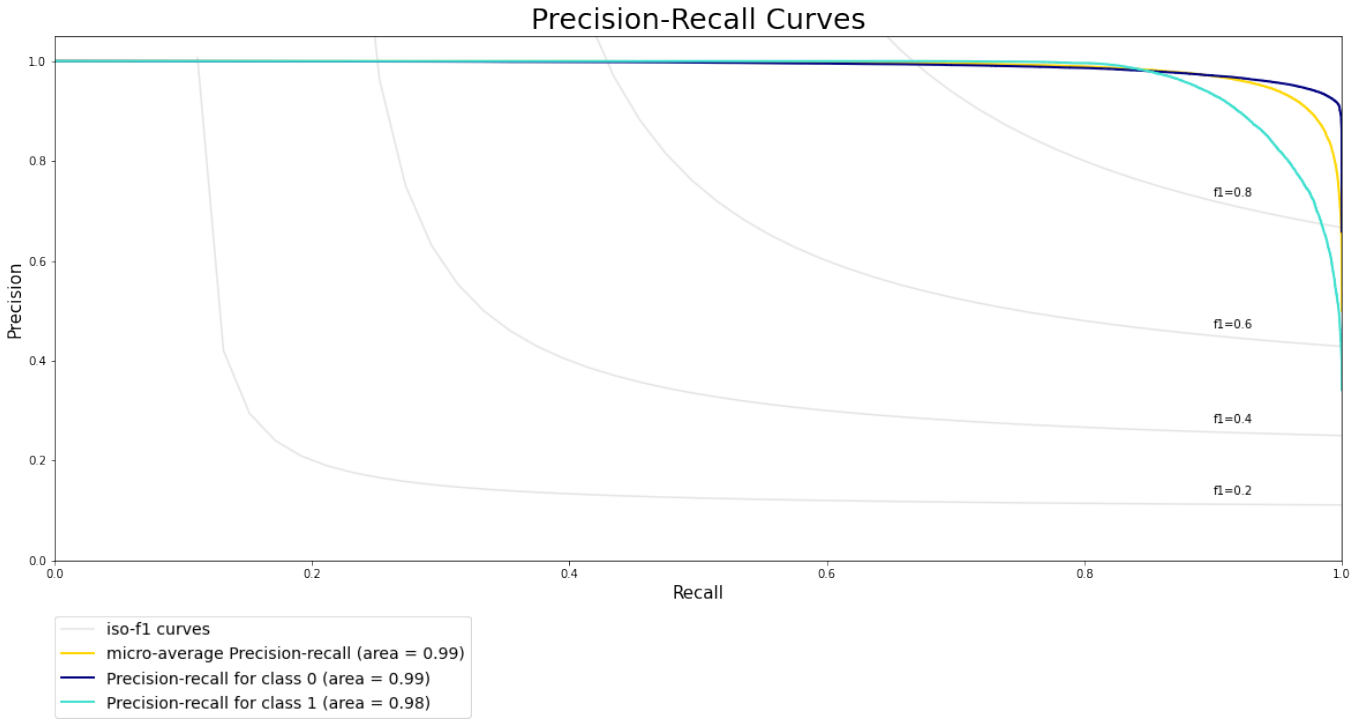

Figure 4.4: Precision-Recall curve

Chapter 5

Conclusion and Future Work

Phishing website attacks pose a serious risk to Internet users, and they have been on the rise recently. Every day and hour, many users unknowingly click phishing URLs. Attackers frequently use phishing because it is simpler to fool a victim into clicking a malicious link that looks authentic than to try to get past a computer's security measures. Many outstanding studies has been conducted to mitigate these threats, but still the detection accuracy is in progress. The goal of this project is to identify and classify URLs into phishing or benign class. The result obtained on the selected metrics: Accuracy, Precision, Recall, and F1score shows that the proposed system has better performance. In the future, this work can be extended to multi-class. Meanwhile, since the residual pipeline has 7 inverted residual blocks, we will try to decrease the number of these blocks.

References

[1] APWG Phishing Activity Trends Report, 4th Quarter 2021, "https://docs.apwg.org/reports/apwg/_ trends_report_q4_2021.pdf"

[2] Aravindhan, R., Shanmugalakshmi, R., Ramya, K., & C, S. (2016). Certain investigation on web application security phishing detection and phishing target discovery. In 2016 3rd international conference on advanced computing and communication systems (pp. 1–10). Coimbatore, India.

[3] Xi Xiao, Dianyan Zhang, Guangwu Hu, Yong Jiang, Shutao Xia, CNN–MHSA: A Convolutional Neural Network and multi-head self-attention combined approach for detecting phishing websites.(2020)

[4] Weiping Wang, Feng Zhang, Xi Luo , and Shigeng Zhang[2] (2019) PDRCNN: Precise Phishing Detection with Recurrent Convolutional Neural Networks, Hindawi Security and Communication Networks Volume 2019, Article ID 2595794, 15 pages, https://doi.org/10.1155/2019/2595794

[5] Zuhair, H., Selamat, A. (2018). Phishing Hybrid Feature-Based Classifier by Using Recursive Features Subset Selection and Machine Learning Algorithms. Recent Trends in Data Science and Soft Computing, 267-277. doi:10.1007/978-3-319-99007-1_26.

[6] Ramesh, G., Gupta, J., & Gamya, P. G. (2017). Identification of phishing webpages and its target domains by analyzing the feign relationship. Journal of Information Security and Applications, 35, 75–84. doi:10.1016/j.jisa.2017.06.001

[7] Cova, M., Kruegel, C., & Vigna, G. (2008). There is no free phish: an analysis of free and live phishing kits. In The 2nd conference on USENIX workshop on offensive technologies (WOOT'08), San Jose, CA, USA.

[8] Liu, W., Liu, G., Qiu, B., & Quan, X. (2012). Antiphishing through phishing target discovery. IEEE Internet Computing, 162, 52–61

[9] Zhang, Y., Hong, J. I.,& Cranor, L. F. (2007). Cantina: a content-based approach to detecting phishing web sites. In Proceedings of the

16th international conference on World Wide Web (WWW '07) (pp. 639–648). Banff, Alberta, Canada

[10] Marchal, S., François, J., State, R., & Engel, T. (2014). Phishstorm: detecting phishing with streaming analytics. IEEE Transactions on Network and Service Management, 11(4), 458–471.

[11] Mohammad, R. M., Thabtah, F., & Mccluskey, L. (2017). Predicting phishing websites based on self-structuring neural network. Neural Computing & Applications, 252, 443–458

[12] Nguyen, L. A. T., Ba, L. T., Nguyen, H. K., & Nguyen, M. H. (2014). An efficient approach for phishing detection using single-layer neural network. In 2014 international conference on advanced technologies for communications (ATC 2014) (pp. 435–440). Hanoi, Vietnam.

[13] Zhang, J., & Li, X. (2017). Phishing detection method based on borderline-smote deep belief network. In G. Wang, M. Atiquzzaman, Z. Yan, & K. K. Choo (Eds.), Lecture notes in computer science: vol. 10658, Security, privacy, and anonymity in computation, communication, and storage. SpaCCS 2017 (pp. 45–53). Cham: Springer.

[14] Verma, R., & Das, A. (2017). What's in a URL fast feature extraction and malicious URL detection. In Proceedings of the 3rd ACM on international workshop on security and privacy analytics (pp. 55–63). Scottsdale, Arizona, USA: ACM.

[15] Yang, P., Zhao, G., & Zeng, P. (2019). Phishing website detection based on multidimensional features driven by deep learning. IEEE Access, 7, 15196–15209.

[16] Maria Sameen, Kyunghyun Han & Seong Oun Hwang (2020).PhishHaven—An Efficient Real-Time AI Phishing URLs Detection System. IEEE Access, Volume 8, 83425 - 83443.

[17] Merwe, A. V. D., Loock, M., Dabrowski, M. (2005). Characteristics and responsibilities involved in a phishing attack. In Winter international symposium on information and communication technologies (pp. 249–254).Trinity College Dublin.

[18] Liang, B., Su, M., You, W., Shi, W., & Yang, G. (2016). Cracking classifiers for evasion: A case study on the google's phishing pages filter. In International conference on World Wide Web (WWW '16) (pp. 345–356). Montréal, Québec, Canada

[19] Abutair, H., Belghith, A., & AlAhmadi, S. J. (2018). Using case-based reasoning forphishing detection. Journal of Ambient Intelligence and Humanized Computing, 1–14

[20] Al-Janabi, M., Quincey, E. D., & Andras, P. (2017). Using supervised machine learning algorithms to detect suspicious URLs in online social networks. In Ieee/Acm international conference on advances in social networks analysis and mining (ASONAM '17) (pp. 1104–1111). Sydney, Australia

[21] Blum, A., Wardman, B., Solorio, T., & Warner, G. (2010). Lexical feature based phishing URL detection using online learning. In Proceedings of the 3rd ACM workshop on Artificial intelligence and security (AISec '10) (pp. 54–60). Chicago, Illinois, USA.

[22] Malicious URLs dataset — Kaggle

[23] Xiao, X., Wang, Z., Li, Q., Xia, S., & Jiang, Y. (2017). Back-propagation neural network on markov chains from system call sequences: a new approach for detecting android malware with system call sequences. IET Information Security, 111, 8–15

[24] Cui, Q., Jourdan, G. V., Bochmann, G. V., Couturier, R., Onut, I. V., & (2017). Tracking phishing attacks over time. In The 26th international conference on World Wide Web (WWW'17) (pp. 667–676). Perth, Australia

[25] Bahnsen, A. C., Bohorquez, C. E., Villegas, S., Vargas, J., & González, F. A. (2017).Classifying phishing URLs using recurrent neural networks. In 2017 APWGsymposium on electronic crime research (eCrime) (pp. 1–8). Scottsdale, AZ, USA.

[26] Ketkar, N. (2017). Convolutional neural networks. Deep Learning with Python, 63–78.

[27] Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. In Computer vision and pattern recognition (pp. 3642–3649). Providence, Rhode Island

[28] Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In International conference on learning representations. San Diego, California

[29] Mnih, V., Heess, N., Graves, A., & Kavukcuoglu, K. (2014). Recurrent models of visual attention. In The 27th international conference

on neural information processing systems (NIPS'14) (pp. 2204–2212). Montreal, Canada.

[30] Luong, M. T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. Computer Science, 1412–1421.

[31] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In 31st conference on neural information processing systems, Long Beach, CA, USA.

[32] Berners-Lee, T., Masinter, L., & McCahill, M. (1994). Uniform resource locators (URL) (pp. 106–107). RFC Editor.

[33] Kim, Y. (2014). Convolutional neural networks for sentence classification. In The 2014 conference on empirical methods in natural language processing (pp. 1746–1751). Doha, Qatarpages

[34] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In 2016 IEEE conference on computer vision and pattern recognition (pp. 770–778). Las Vegas, NV, USA