Ministry of Higher Education and Scientific Research

وزارة التعليم العالي والبحث العلمي

Badji Mokhtar Annaba University
Université Badji Mokhtar – Annaba
Faculty of Technology

Department of Computer Science

جامعة باجي مختار – عنابـــة

كـليـة التكنـولـوجيـا

قسـم الاعلام الالـي

Thesis

Presented to obtain the diploma of

Doctorate (3rd cycle)

Sector: Computer Science

Speciality: Networks and Security

By :

ZAIMI Rania

Theme :

Protection de la vie privée des utilisateurs sur le web

Defended on 23-01-2025 in front of the jury composed of:

N°	First and last name	Grade	Establishment	Quality
01	Ghoualmi-Zine Nacira	Prof	Badji Mokhtar –Annaba University	Chair
02	Hafidi Mohamed	Prof	Badji Mokhtar –Annaba University	Supervisor
03	Azizi Nabiha	Prof	Badji Mokhtar –Annaba University	Examiner
04	Zarzour Hafed	Prof	Mohamed-Chérif Messaadia – Souk Ahras University	Examiner
05	Bey Anis	MCA	ESSG-Annaba	Examiner

Acknowledgments

I would like to express my acknowledgments and gratitude, first and foremost, to Almighty "Allah" for granting me the courage, strength, and determination to accomplish this work.

I want to thank my mother and my entire family for their support and encouragement, without which none of this would have been possible.

I extend my appreciation to the members of the jury, Professor. Zarzour Hafed, Professor. Azizi Nabiha and Dr Bey Anis for agreeing to be examiners for this thesis. I am equally thankful to the jury president, Professor. Ghoualmi-Zine Nacira for accepting the presidency role.

I would also like to express my acknowledgment to my supervisor, Professor. Hafidi Mohamed, and also to Professor. Mahnane Lamia for all their assistance, guidance, and encouragement.

Special thanks go to my friends and colleagues at the LRS laboratory and the Department of Computer Science, and to everyone who has directly or indirectly contributed to the realization of this project.

****Dedication*****

I dedicate this thesis to the memory of my father, who always encouraged me in my studies and pushed me to achieve a higher level in higher education. I hope he will be proud of me wherever he is.

I dedicate this thesis to my mother, whose unwavering support, encouragement, and sacrifices have been the driving force behind my academic journey.

This achievement is as much theirs as it is mine.

I extend this dedication to my entire family, from the youngest to the oldest, and to all those who believed in me and are happy and proud of my success.

Rania

الملخص

في الآونة الأخيرة، أصبح الأفراد في جميع أنحاء العالم أكثر اعتمادًا على التكنولوجيا، ولا سيما الإنترنت. هذا التحول في الاعتماد قد سهّل بشكل كبير الحياة اليومية، مما أدى إلى زيادة ملحوظة في استخدام منصات التجارة الإلكترونية والمعاملات الرقمية. هذه الاتجاهات كانت بارزة بشكل خاص منذ بدء جائحة كوفيد١٩، حيث تم تقييد الوسائل التقليدية، مما دفع إلى انتشار واسع للعمل عن بعد والاجتماعات الافتراضية كممارسات قياسية. ونتيجة لذلك، فإن الكم الهائل من البيانات الشخصية المتداولة عبر الإنترنت يشكل تهديدًا كبيرًا لخصوصية المستخدمين. تزيد العديد من التهديدات السيبرانية، بما في ذلك الهجمات السيبرانية، على هذا الوضع. تشير العديد من المصادر إلى زيادة كبيرة في الهجمات السيبرانية منذ ظهور جائحة كوفيد١٩، مما يسلط الضوء على الترابط بين الأمان السيبراني والخصوصية على الإنترنت ضمن المجال الأوسع للأمان الرقمي وحماية الخصوصية.

هدف أطروحتنا هو إجراء دراسة حول حماية الخصوصية على الإنترنت. اخترنا تضييق هذه المسألة من خلال التركيز على التحدي الخاص بحماية البيانات الشخصية من التصيد الاحتيالي، وهو أكثر أنواع الهجمات شيوعًا في مجال الأمن السيبراني لسرقة الهوية. يعد التصيد الاحتيالي نشاطًا احتياليًا يُحث فيه الأفراد والمنظمات على زيارة عناوين URL الضارة ومشاركة بياناتهم الشخصية، بما في ذلك كلمات المرور وتفاصيل بطاقات الائتمان ومعلوماتهم الشخصية الأخرى، مما يؤثر مباشرة على

خصوصيتهم على الإنترنت. تُنفَّذ هذه الهجمات الخداعية بشكل استراتيجي لخداع المستخدمين وجعلهم يعتقدون أنهم يتفاعلون مع موقع ويب أو خدمة عبر الإنترنت موثوقة لسرقة معلومات حسابهم لأغراض ضارة.

تم تطوير العديد من الحلول المضادة للتصيد الاحتيالي في الأدب، ومع ذلك، فإن المهاجمين يبتكرون باستمرار حيل جديدة لتفادي هذه التقنيات، مما يؤدي إلى زيادة في إحصائيات هجمات التصيد الاحتيالي مع مرور الوقت. وهذا يؤكد ضرورة مواصلة البحث في مجال التصيد الاحتيالي. تركز هذه الأطروحة على إنشاء نظام قوي للكشف عن مواقع الويب الخاصة بالتصيد الاحتيالي بناءً على التقنيات الناشئة مثل التعلم العميق ومعالجة اللغة الطبيعية (NLP) . تقدم الحلول المقدمة رؤى إضافية مقارنة بالبحوث السابقة في الأدب، مما يحسن من كفاءة الكشف ويساهم في الأمن السيبراني.

• **الكلمات المفتاحية:** حماية الخصوصية على الإنترنت، كشف مواقع التصيّد، التعلم العميق، التعلم الآلي، الأمن السيبراني

Résumé

Dans les temps récents, les individus à travers le monde sont devenus de plus en plus dépendants de la technologie, notamment d'Internet. Ce changement de dépendance a considérablement simplifié les activités quotidiennes, entraînant une forte augmentation de l'utilisation des plateformes de commerce électronique et des transactions numériques. Cette tendance s'est particulièrement accentuée depuis l'avènement de la pandémie de COVID-19, pendant laquelle les méthodes traditionnelles étaient limitées, incitant à une adoption généralisée du travail à distance et des réunions virtuelles comme pratiques standard. En conséquence, la masse importante de données personnelles circulant en ligne pose une menace significative pour la vie privée des utilisateurs. De nombreuses menaces cybernétiques, y compris les cyberattaques, exacerbent cette situation. De nombreuses sources indiquent une augmentation substantielle des cyberattaques depuis l'émergence de la pandémie de COVID-19, mettant en lumière l'interconnexion de la cybersécurité et de la confidentialité en ligne dans le domaine plus vaste de la sécurité numérique et de la protection de la vie privée.
Notre thèse vise à mener une étude sur la protection de la vie privée sur Internet. Nous avons choisi de restreindre cette problématique en nous concentrant sur le défi de la protection des données personnelles contre le phishing, le type d'attaque le plus répandu dans le domaine de la cybersécurité pour le vol d'identité. Le phishing est une activité frauduleuse au cours de laquelle des individus et des organisations sont incités à visiter des URL malveillantes et à partager leurs données personnelles, notamment leurs mots de passe, les détails de leur carte de crédit et d'autres informations personnelles, impactant directement leur vie privée en ligne. Ces attaques trompeuses sont stratégiquement exécutées pour tromper les utilisateurs en leur faisant croire qu'ils interagissent avec un site Web ou un service en ligne digne de confiance afin de voler les informations de leur compte à des fins malveillantes.

De nombreuses solutions anti-phishing ont été introduites dans la littérature, mais les attaquants continuent de concevoir de nouveaux stratagèmes pour contourner ces techniques, ce qui se traduit par une augmentation des statistiques d'attaques de phishing au fil du temps. Cela souligne la nécessité de poursuivre la recherche dans le domaine du phishing. Cette thèse se con-

centre sur l'établissement d'un système robuste de détection de sites Web de phishing basé sur des technologies émergentes telles que l'apprentissage approfondi et le traitement du langage naturel (NLP). Les solutions introduites fournissent des informations supplémentaires par rapport aux recherches antérieures de la littérature, améliorant ainsi l'efficacité de la détection et contribuant à la cybersécurité.

- **Mots Clés:** Protection de la vie privée en ligne, Détection de sites web de phishing, Apprentissage profond, Apprentissage automatique, Cyber-sécurité.

Abstract

In recent times, people across the globe have grown more reliant on technology, notably the internet. This shift in reliance has notably streamlined everyday activities, resulting in a notable surge in the utilization of e-commerce platforms and digital transactions. This trend has been particularly pronounced since the advent of the COVID-19 pandemic, during which conventional methods were constrained, prompting a widespread adoption of remote work and virtual gatherings as standard practices. Consequently, the vast amount of personal data circulating online poses a significant threat to users' privacy. Numerous cyber threats, including cyberattacks, exacerbate this. Many sources indicate a substantial increase in cyberattacks since the emergence of the COVID-19 pandemic, highlighting the interconnectedness of cybersecurity and online privacy within the broader domain of digital security and privacy protection.

Our thesis aims to conduct a study on privacy protection on the Internet. We have chosen to narrow this issue by focusing on the challenge of safeguarding personal data against Phishing, the most prevalent type of attack in the cybersecurity realm for identity theft. Phishing is a fraudulent activity in which individuals and organizations are lured to visit malicious URLs and share their personal data, including passwords, credit card details, and other personal information, directly impacting their online privacy. These deceptive attacks are strategically executed to deceive users into believing they are interacting with a trustworthy website or online service to steal their account information for malicious purposes.

Numerous anti-phishing solutions have been introduced in the literature, yet attackers continuously devise new tricks to bypass these techniques, resulting in an increase in phishing attack statistics over time. This underscores the necessity to continue researching the phishing domain. This thesis focuses on establishing a robust phishing website detection system based on emerging technologies such as deep learning and NLP (Natural Language Processing). The solutions introduced provide additional insights compared to previous literature research, improving detection efficiency and contributing to cybersecurity.

• **Keywords:** Protection of Online Privacy, Phishing Websites Detection, Deep Learning, Machine Learning, Cyber-Security.

Table of Contents

List of Figures

List of Tables

List of Acronyms & Abbreviations

1D One Dimensional. iv, 96, 97

2D Two Dimensional. iv, 96, 101

AI Artificial Intelligence. 45, 47

ALBERT A Lite BERT. 139

ANN Artificial Neural Network. 58

APWG Anti-Phishing Working Group. 13, 19, 36

AUC area under the curve. 52, 54, 56, 57

BERT Bidirectional Encoder Representations from Transformers. 14, 134, 138

BSSA Binary Slap Swarm Optimization Algorithm. 52, 58, 131

CEO Chief Executive Officers. 19, 40

CGI Consultants to Government and Industry. 19

CNN Convolutional Neural Network. iv, 15, 47, 84, 86, 92, 96, 97, 101, 107, 116

COVID-19 Coronavirus Disease 2019. 16, 18

CSS Cascading Style Sheets. 78

CSV Comma-Separated Values. 71

DDoS Distributed Denial of Service. ii, 27, 28

DES Data Encryption Standard. 32

DistilBERT Distilled BERT. 14, 15, 133, 139

DL Deep Learning. iv, v, 13, 15, 16, 47, 67, 69, 70, 83, 90, 107

DNS Domain Name System. iii, 28, 29, 43, 48

DoS Denial of Service. ii, 16, 27, 32, 42

DT Decision Tree. 52, 53, 105

ELECTRA Efficiently Learning an Encoder that Classifies Token Replacements Accurately. 139

FN False Negative. 89

FP False Positive. 89

FPR False Positive Rate. 54

GDPR General Data Protection Regulation. 1, 10, 11

GNB Gaussian Naïve Bayes. 52, 105

HTTPS Hypertext Transfer Protocol Secure. 71

IDF Inverse Document Frequency. 53

IDS Intrusion Detection Systems. 30

iOS iPhone Operating System. 28

IoT Internet of Things. ii, 11, 22, 28

IT Information Technology. ii, 24

KNN K Nearest Neighbors. 52, 53, 56, 57, 105

LG Logistic Regression. 53

LightGBM Light Gradient Boosting Machine. 53

LL Linear Learner. 105

LSTM Long Short-Term Memory. 15, 47, 107, 118

MitM Man in the middle attack. 27

ML Machine Learning. v, 13, 47, 67, 107

MLM Masked Language Model. 138

MLP Multilayer Perceptron. 53

NB Naïve Bayes. 53

NLP Natural language processing. 14, 15, 138, 140

NSP Next Sentence Prediction. 138

PIM permutation importance method. 5, 15, 107, 114, 131, 132

ReLU Rectified Linear Unit. 85

RF Random Forest. 52, 105

RNN Recurrent Neural Network. 47, 87

RoBERTa Robustly optimized BERT approach. 139

RSA Rivest-Shamir-Adleman. 32

SAAS software-as-a-service. 36

SMOTE Synthetic Minority Over-sampling Technique. 109

Smote-Tomek link Synthetic Minority Over-sampling Technique Tomek link. 15, 108–110, 138

SMOTEENN Synthetic Minority Over-sampling Technique plus Edited Nearest Neighbors. 109

SMS Short Message Service. 41

SPF Sender Policy Framework. 82

SQL Structured Query Language. iii, 28

SSL Secure Sockets Layer. 82

TLS Transport Layer Security. 82

TN True Negative. 89

TP True Positive. 89, 90

TPR True Positive Rate. 54

WCA Water Cycle Algorithm. 52

WEF World Economic Forum. 19

WLAN Wireless Local Area Network. 42

WOT Web of Trust. 48

XGBoost Extreme Gradient Boosting. 15, 52, 57, 105, 107, 115

Chapter. 1

Introduction

1.1 Context

The Internet has played an increasingly significant role in people's work and lives in recent years. Since its inception, it has succeeded in gathering a vast amount of the world's knowledge in a centralized location that everyone can access. After a few decades, this technology has become a major player in communications, commerce, and social relationships. Recent innovations in information and communication technologies, including the widespread availability of smart devices and wireless sensor networks, have led to a significant increase in the volume of information circulating on the Internet. This information has rapidly become the new currency.
Consequently, today's Internet is filled with businesses and organizations whose sole purpose is to gather as much information as possible. This technology has become a major factor in communications, commerce, and social relationships, leading to a significant increase in the volume of information circulating on the Internet. This accumulation of information can be used outside of its legitimate context, directly impacting people's online privacy. Several key online privacy principles must be adhered to when managing personal digital data. The GDPR delineates six data protection principles, as illustrated in Figure 1.1. Firstly, according to the principles of Lawfulness, fairness, and transparency, personal data must be processed legally, fairly, and transparently concerning the data subject. Secondly, the principle of Purpose limitations dictates that personal data should only be collected and used for specific, explicit, and legitimate purposes of which the subject has been made aware, and not for any other purposes. Thirdly, Data minimization ensures that the personal data processed is adequate, relevant, and limited to what is necessary for the intended processing purpose. Fourthly, data must be accurate and, where necessary, kept up to date. Fifthly, under Storage limitation, personal data should not be retained for longer than necessary. Finally, Integrity and confidentiality require measures to guarantee the security of individual information, counting assurance against illegal handling, coincidental misfortune, destruction, or harm. [38], [132].

Figure 1.1: Privacy principles for GDPR [38].

The protection of a user's online privacy is closely linked to their security both online and offline. Breaking the confidentiality principle directly impacts online privacy. As shown in Figure 1.2, there are numerous threats to privacy, including weak or reused passwords, oversharing on social media, the widespread adoption of IoT devices that constantly collect data about users, unsecured web browsing, and software vulnerabilities. All of these factors make cybercriminals eager to exploit them and carry out cyberattacks, such as collecting data without the user's consent (e.g., IP address, location) and identity theft through phishing, which compromises the security of personal data on the web [23].

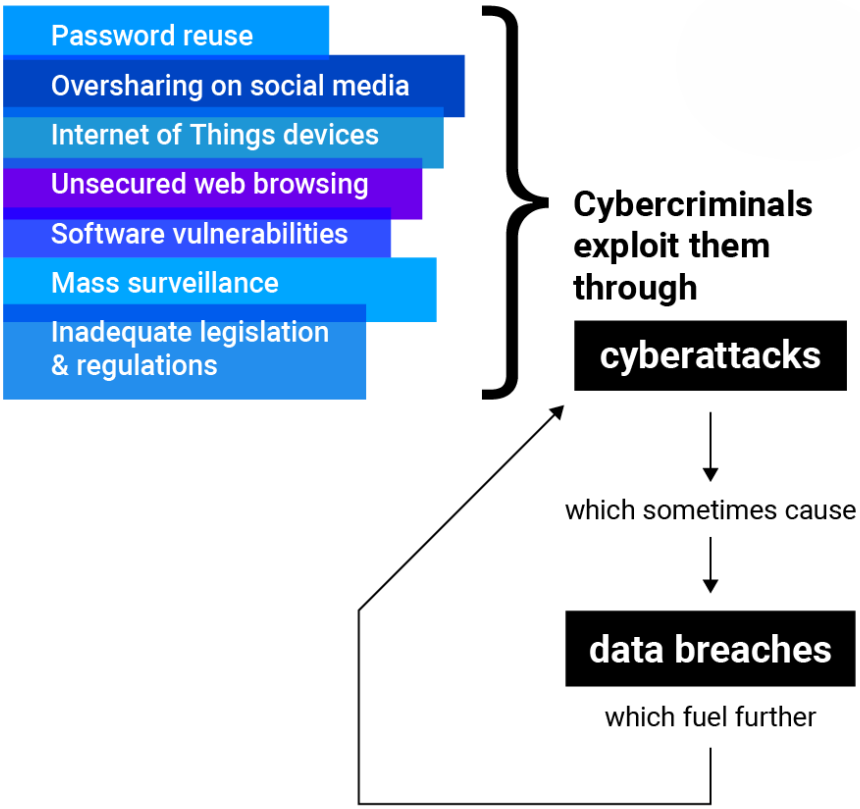


Figure 1.2: Threats to online privacy [23].

1.2 Motivation and Problematic

With the rapid development of databases, the internet, and information technology, a vast amount of personal data can be integrated and digitally analyzed. It is crucial to understand that any unsecured information posted online can be accessible to everyone. Our thesis aims to study the protection of user privacy on the web. We have chosen to focus on the security principle of online privacy, which is affected by various cyberattacks threatening user privacy. We cannot address all cyberattacks threatening internet users' privacy. Therefore, we have chosen to address phishing attacks, a type of social engineering attack that compromises the privacy of web users. Phishing is defined as the fraudulent act of acquiring private and sensitive data by posing as a trusted entity. Typically, the targeted data includes login credentials, passwords, credit card numbers, or social security numbers. Phishing attacks are the most prevalent in the cybersecurity domain and are widely used on the internet for identity theft. The quantity of phishing attacks has recently increased significantly and is considered one of the most perilous violations of the web today. Despite existing tools, the number of victims continues to rise. The security challenges posed

by phishing are rapidly escalating. According to the Anti-Phishing Working Group (APWG), in the second quarter of 2023, were 1,286,208 observed phishing attacks, marking the third-highest quarterly total ever recorded by the APWG [13]. The harmful effects of phishing on victims could extend to accessing users' confidential information, resulting in financial losses and even preventing them from accessing their accounts, leading people to lose confidence in online businesses regarding their web security. Detecting and stopping malicious websites have become crucial control measures to avoid information security risks. Phishing attacks can be prevented by detecting malicious websites and raising users' awareness of identifying phishing websites. Despite ongoing research efforts worldwide to prevent phishing attacks, it remains the most active, current, and widespread attack in the cybersecurity domain and continues to grow. This emphasizes the need to develop an effective approach to detecting phishing websites based on emerging technologies. Such contributions are invaluable in the field of cybersecurity, aiding in the protection of users' sensitive web data.

Numerous methods for detecting and preventing phishing attacks through software have been explored in the literature, including list-based, visual similarity, heuristics-based approaches, and Artificial Intelligence. Each of these approaches carries its own set of advantages and drawbacks. For instance, while the blacklist approach may overlook phishing sites not yet identified (known as zero-day attacks), visual similarity techniques based on image analysis can be resource-intensive in terms of time and space. Recently, baseline machine learning and deep learning approaches in phishing detection have been significantly adopted, driven by rapid advancements and numerous successful studies. However, deep learning methods overshadow the challenge of identifying intricate structures within high-dimensional data, surpassing the manual feature extraction issue encountered in traditional machine learning approaches. Several mechanisms for detecting phishing websites based on ML and DL classifiers have been proposed in the literature. However, most of the proposed mechanisms face various detection shortcomings in several aspects such as:

- Databases: the choice and type of databases used (data imbalance, types of extracted features not covering all aspects of the website (URL, content...etc), the number of samples.

- Data Preprocessing: the choice of the feature selection method, the choice of data balancing method.

- Classifiers: the choice of classification models and their suitability with the classification domain, the proposed architecture of deep learning for classification, training time, detection response time...etc.

- Optimization: this concerns the techniques used to optimize the parameter values of the classification model to achieve optimal classification.

- Performance metrics: this concerns the relevant choice of evaluation metrics to determine that the detection results obtained are reliable.

- User-friendliness: if this phishing detection system is implemented in an easy-to-use manner, it will provide as a final product to web users, such as web applications with interfaces or web browser extensions, assistance even for those who are not experts in the field of protecting against this attack.

1.3 Objective

The primary objective of this thesis is to safeguard web users against phishing attacks by proposing an efficient phishing website detection system utilizing emerging technologies. Through our research, we aim to develop a robust system capable of ensuring a high detection rate and a low false positive rate for efficient and reliable classification. Additionally, we address the feasibility of our proposed phishing detection system, considering tactics employed by phishers, such as content imitation and URL tricks, to enhance its effectiveness. We then proposed security solutions aimed at ensuring the protection of web users against phishing attacks. These solutions consist of mechanisms for detecting phishing websites based on the following emerging technologies:

- Deep learning, which aims to cover all the aspects mentioned above (choice of databases, preprocessing techniques, classification models, etc.), and to perform efficient and reliable classification.

- Natural language processing (NLP), which analyzes phishing URLs without extracting characteristics, to process the complex structure of URL data. Particularly, this technology is not widely exploited in the field of phishing detection for a more innovative and recent detection mechanism.

1.4 Thesis contributions

The main contribution of this thesis aims to establish a robust phishing website detection system based on emerging technologies, offering additional insights compared to previous literature research concerning the following points:

- Databases: The utilization of five robust databases covering all types of features to address phishers' tricks such as URL and web content manipulation, ensuring effective detection.

- Preprocessing: The adoption of new preprocessing techniques such as the permutation importance feature selection method to select the most relevant characteristics, and the SMOTE-Tomek link data balancing method to balance imbalanced databases by considering oversampling and undersampling without significant loss of important samples. The application of the Distilled BERT (DistilBERT) model, which is a distilled version of Bidirectional Encoder Representations from Transformers (BERT), for extracting meaningful text information from URLs.

- Classification: The selection of the most suitable deep learning models in this domain is based on state-of-the-art studies, including CNN, Long Short-Term Memory (LSTM), and a hybrid model. The hybrid model is chosen to combine the advantages of each classifier and overcome their limitations for improved detection.

- Optimization: Employing a parameter tuning process to optimize the deep learning models' parameters, ensuring optimal classification performance.

- Performance Metrics: Exploring the most appropriate performance metrics in this domain to assess the reliability of the obtained results.

The contributions of this thesis can be summarized as follows:

- An effective phishing website detection system utilizing CNN is developed, testing both 1D CNN and 2D CNN architectures with three types of features: URL-based features, content-based features, and third-party services-based features. The proposed system is capable of classifying websites as either legitimate or phishing. Additionally, a series of experiments are conducted to optimize parameters for each CNN model, assessing the impact of the parameter tuning process on the deep learning model's performance. The results obtained demonstrate the successful operation of the proposed system, achieving high accuracy values. Moreover, compared to existing literature, the proposed system shows an improvement in accuracy and reduces training time.

- An intelligent, deep-learning-based mechanism for detecting phishing URLs using the PIM to select the most relevant URL features and the Synthetic Minority Over-sampling Technique Tomek link (Smote-Tomek link) method to address the issue of an unbalanced dataset. Additionally, the Extreme Gradient Boosting (XGBoost) and four DL models—CNN, LSTM, and two hybrid models (CNN-LSTM and LSTM-CNN)—are tested to assess the potential of DL models and identify the most suitable detection model for phishing. The experimental results demonstrate the successful operation of the proposed phishing detection mechanism, outperforming existing literature that utilized the same dataset in terms of accuracy as a performance metric. Furthermore, our phishing detection mechanism is implemented as a web application to enhance its usability for web users.

- A malicious URL detection mechanism based on the natural language processing model DistilBERT is utilized for feature extraction from URLs in terms of meaningful text information. Additionally, a hybrid deep learning model (CNN-LSTM) is employed to classify the dataset into malicious and benign URLs. Based on the experimental results, it is demonstrated that the combination of NLP and DL yielded successful outcomes in this field, achieving significantly high accuracy values that surpass existing literature.

1.5 Structure of the thesis

After the first chapter, which serves as a general introduction, the thesis is comprised of the following chapters:

The chapter 2 provides an overview of online privacy and Cybersecurity, detailing key concepts related to privacy. In particular, we introduce the definition and impact of the Coronavirus Disease 2019 (COVID-19) pandemic on internet privacy protection. We also provide an overview of Algerian legislation that governs this area and mention various principles and threats to online privacy. Additionally, we focus on digital security by defining information security, online privacy, cybersecurity, and cybercrime. Furthermore, we discuss common types of cyberattacks such as malware, phishing, scams, man-in-the-middle attacks, DoS attacks, etc. Finally, we discuss the main systems and tools used to combat these attacks.

Chapter 3 provides an extensive review of the literature on phishing, elucidating fundamental concepts including the definition of phishing attacks, motivations, statistics, various attack types, diverse anti-phishing solutions, future directions, and relevant studies to deepen comprehension of this domain.

Chapter 4 presents the fundamental elements and terminology necessary for understanding DL in the field of phishing detection. Furthermore, it highlights our contributions to cybersecurity in the context of applying DL to detect phishing websites.

Chapter 5 introduces an enhanced approach for detecting malicious URLs utilizing the DistilBERT method for feature extraction and Deep Learning for classification. A hybrid model (CNN-LSTM) is used to classify the dataset into malicious and benign URLs.

Finally, Chapter 6 concludes the manuscript and summarizes what has been accomplished in the thesis. Additionally, it offers an overview of the perspectives and future research in the field.

Chapter. 2

Online Privacy & Cybersecurity: Background and Basic Concepts

2.1 Introduction

In the present day, technology's swift emergence has led individuals to increasingly rely on connected devices like computers and smartphones. They engage in various online activities, including e-commerce, bill payment, online shopping, and social network use. However, this extensive connectivity compels individuals to exchange sensitive personal data, putting their privacy at risk and making them susceptible to cyberattacks. The crux of user privacy on the internet lies in users' ability to maintain transparency and control over the collection and use of their data, whether by private or public entities. This chapter aims to provide an overview of online privacy and cybersecurity, delving into fundamental concepts such as principles, cyberattacks that pose threats to them, and strategies for ensuring robust cybersecurity measures.

2.2 Online privacy definition

Online privacy is defined by how much an individual's personal information is safeguarded while using the internet. This encompasses the level of security available for online communications, preferences, as well as personal and financial data. Common strategies employed by internet users to enhance their online privacy include utilizing antivirus software, selecting strong passwords, disabling tracking features, verifying website security, and opting for stricter privacy settings. However, online privacy faces threats such as malware and phishing schemes, along with potential vulnerabilities in website security that could result in identity theft [122].

2.3 The Impact of the COVID-19 Pandemic on Online Privacy & Cybersecurity

The COVID-19 contagion has led to an increased reliance on online communication and technology, as people have been forced to work, study, and socialize from home. This has raised concerns about online privacy, as individuals are sharing more personal information and conducting more activities online. With more people using online platforms, there has been a corresponding increase in cyberattacks, scams, and phishing attempts that exploit the pandemic for malicious purposes. Furthermore, the collection and use of personal data by governments and private companies for COVID-19-related purposes, such as contact tracing and health monitoring, has also raised privacy concerns. Since the covid 19 pandemic, organizations have discovered a new way of operating by embracing remote and hybrid work arrangements, this working culture is still adopted to this day, therefore, there is an increased need for greater emphasis on cybersecurity, as there is a higher risk of cyber threats. This is evidenced by the fact that nearly half of all individuals succumb to phishing scams while working from home. Opportunistic cyber attackers are taking advantage of the pandemic to increase their criminal activities by exploiting the vulnerabilities of remote employees, including their heightened interest in coronavirus-related news, which can lead to the spread of malicious websites. Additionally, it is worth noting that the average cost of a data breach that occurs as a result of remote work can reach up to $137,000 [93]. One instance of cybercriminals taking advantage of the cybersecurity vulnerabilities that arise with remote work is the sequence of cyberattacks that targeted video conferencing services. During the period of February 2020 through May 2020, over 500,000 individuals were impacted by breaches that involved the theft and sale of personal information, such as usernames, passwords, and email addresses, of users of video conferencing services on the dark web. To carry out these attacks, some hackers utilized a tool known as 'OpenBullet' [93]. Before the onset of the pandemic, around 20% of cyberattacks employed techniques or malware that had not been previously encountered. However, since the pandemic began, this proportion has increased to 35%. Some of the new attacks use machine learning technology that can adapt to its surroundings, making it harder to detect. For instance, phishing attacks are becoming more advanced, with attackers utilizing alternative channels like SMS and voice (vishing). Additionally, attackers are using news related to vaccine advancements to carry out phishing campaigns. Furthermore, ransomware attacks are also becoming more sophisticated, with hackers combining data leakage attacks with ransomware to coerce victims into paying the ransom [93]. More statistics about evidence increase in attacks during the COVID-19 period are provided in Table 2.1 [79, 56], [65]. Due to all these statistics, individuals need to be aware of potential online privacy risks and take necessary steps to protect themselves.

Table 2.1: Statistics on Cyber-attacks Arising from the COVID-19 Pandemic Outbreak.

Ref Source	Statistic
[79, 117]	There was a recorded increase of 600% in phishing attempts in March 2020.

Continued on next page

Table 2.1 – continued from previous page

Source	Statistic
[79, 55]	The World Economic Forum (WEF) recorded a 50.1% increase in cyber attacks during the pandemic, with around 30,000 cyber incidents linked directly to COVID-19 from December 31, 2019, to April 14, 2020.
[79, 86]	The Consultants to Government and Industry (CGI) group reported a remarkable 30,000% increase in cyber threats directly linked to COVID-19. Moreover, CGI's Security Operations Centers noted a corresponding spike, with a significant 30,000% surge in threats associated with COVID-19, including malware, weaponized websites, and phishing emails.
[79, 42]	From January to April 2020, Interpol detected roughly "907,000 spam messages, 737 instances of malware, and 48,000 malicious URLs linked to COVID-19." They also noted that the "average ransomware payment in the second quarter of 2020 reached $178,254, indicating a 60% increase compared to the first quarter."
[79, 77]	During April 2020, Google was said to have been halting around 18 million malware and phishing emails linked to the virus every day.
[65, 11], [65, 12]	There has occurred a significant increase, particularly in phishing endeavors, which quadrupled during the pandemic as per the APWG (2020), and has subsequently increased eightfold as per APWG (2022).
[65, 10]	Cyber attackers have exploited the COVID-19 crisis by extensively creating phishing emails. Media reports have frequently highlighted common examples such as Zoom phishing emails, fraudulent Chief Executive Officers (CEO) emails, and phishing attempts aimed at healthcare institutions.

Continued on next page

Table 2.1 – continued from previous page

Source	Statistic
[65, 4]	The sharp rise in COVID-19 phishing scams worldwide can be linked to the outbreak. The upheaval resulting from the disaster leaves society more open to fraudulent activities, leading to increased susceptibility to phishing attempts.

2.4 Online Privacy & Legislation

The unauthorized gathering, utilization, and distribution of personal data with external entities, without informing or obtaining permission from users, is also a matter of worry. As many as 137 countries out of 194 have introduced regulations to ensure the safeguarding of data and privacy. However, Africa and Asia exhibit distinct degrees of implementation, with only 61% and 57% of countries having implemented such legislation, respectively. In the least developed countries, the percentage is even lower, at only 48% [128]. Recently, Algeria began to care about the protection of privacy on the Internet. A self-governing administrative body, commonly referred to as the "national authority," has been established and is based in Algiers to protect personal data. The national authority is responsible for ensuring that personal data processing is conducted according to legal regulations and that the utilization of information and communication technologies does not jeopardize the rights of individuals, public freedoms, and privacy. Despite Law No. 18-07 mandating the establishment of a national authority, it has yet to be established. The law aims to protect the privacy of individuals about the processing of their data, it also provides individuals with the right to access their data and to have their data deleted or corrected if necessary. It also requires data controllers to obtain the consent of individuals before processing their data and to inform them of the purpose of data processing. Nonetheless, Presidential Decree No. 22-187 issued on May 18, 2022, has appointed the President and members of the National Authority for Personal Data Protection. Although the national authority is still not accessible to the public, it is expected to become operational soon [99].

2.5 The principles of privacy on the Internet

2.5.1 Authority to collect

The collection of personal information should be limited to institutions with the use of authorized activities [95]. That's to say, the reason(s) for collecting personal information, the legal basis, and the name and contact information of the person who can respond to inquiries about the collection must be specified [37].

2.5.2 *Mode of collection*

It is necessary to ensure that personal data is only collected directly from people unless there are a few exceptional circumstances. According to the law, personal data must be obtained directly from the person concerned [95].

2.5.3 *Transparency & Notice requirements (Consent)*

It is necessary to inform an individual of the collection or any secondary use of their personal information [95].

2.5.4 *Limited Use and Disclosure*

This principle requires that the use and disclosure of personal information must be limited to authorized activities and for the purposes for which they are collected [95], [37], [19].

2.5.5 *Accuracy*

Necessary processes must be made to ensure that the personal information collected is accurate and complete when used to make decisions affecting data subjects [95], [37].

2.5.6 *Data Sovereignty (right of access)*

For a limited time, people have the right to access their personal information, review it, determine whether it is accurate, complete, or outdated, and request that it be deleted [95], [37], [19].

2.5.7 *Security*

This principle must ensure the confidentiality of personal information and security measures against loss, unauthorized access, misuse, or disclosure [95], [37], [19].

2.5.8 *Limit Retention*

Ensure that the transfer of personal information to archives and its destruction is authorized, secure, and carried out following any applicable records retention schedules [95], [37], [19].

2.6 Threats to privacy

The most common threats to privacy are depicted in Figure 2.1.

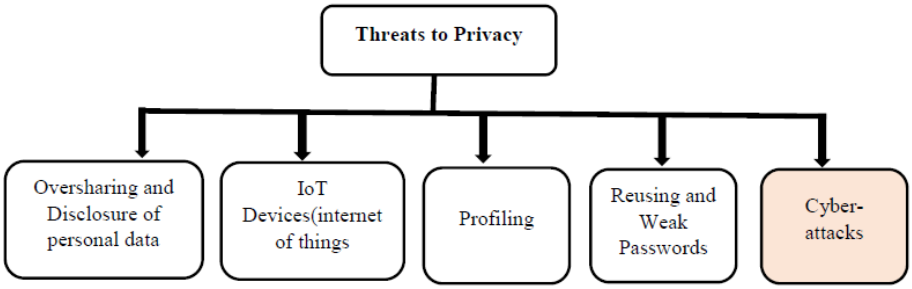


Figure 2.1: Threats to Privacy.

2.6.1 Oversharing and disclosure of personal data

With the emergence of social networks, photo, and video-sharing services, and other technological advancements, it is now incredibly simple to share every aspect of our lives to strengthen our social connections. Information like Date of birth, place of employment, marital status, musical tastes, behavioral traits, and other details that the majority of us consider unimportant and harmless can be used against us and reveal more personal information about us than we would ever want to share. we also paint a clearer picture of our lives, routines, significant connections, and possessions with each post. For instance, sharing images of our homes or places of employment makes it easier for others to locate us, which in some situations can lead to awkward situations. Moreover, the risks of prejudice, inequality, discrimination, and loss of autonomy are further made apparent by this sensitive information [19], [22].

2.6.2 IoT Devices

Online privacy is also at risk from all the devices connected to the Internet that are constantly listening, recording, and gathering information. The risks to online privacy are rising as a result of the ease of use and constant improvement of IoT devices. For years, individuals have generated personal data such as shopping lists, body temperatures, and fridge contents, yet it has never piqued anyone's curiosity. Nowadays, connected TVs, toasters, and toothbrushes are everywhere [22].

2.6.3 Profiling

Profiling refers to the automated analysis of individualized information to evaluate various characteristics associated with a person, including IP addresses, analyzing and predicting a person's financial situation, spending patterns, search engine queries, websites visited, relationships, and social network communications, as well as preferences, interests, behavior, and more. Profiling is considered a privacy breach if it is used without the individual's consent, such as when data is illegally collected using various tracking and surveillance techniques, or if the profiles created by the processing of collected data are used for bad purposes. However, recommender systems, which offer customers products and services that match their preferences and interests, are examples of beneficial uses for profiling [19], [97], [126], [32], [20].

2.6.4 Reusing and weak passwords

One of the primary causes behind the notable data breaches frequently covered in the media stems from the repeated use of easily guessed passwords. This practice allows cyber attackers to gain access to numerous accounts simultaneously, facilitating activities such as financial fraud and identity theft, often both at once [22].

2.6.5 Cyber-attacks

Cyber attackers can readily target those who depend solely on default settings and employ overly simplistic passwords. To ensnare individuals into clicking, downloading, and opening their traps, malicious hackers blend technical skills with psychological manipulation, enticing them to click links in deceptive messages sent via phone (smishing), open misleading emails (phishing), or even disclose personal information during fraudulent phone calls (vishing). Keyloggers, malicious programs crafted to record all keystrokes on a device and collect usernames and passwords, are frequently employed in such attacks. Subsequently, the acquired data may be utilized for various nefarious purposes, including theft, unauthorized access to private information, or causing general distress. [22].

2.7 Digital Security: Online Privacy, Information Security, and Cybersecurity

Information security guarantees the availability, integrity, and confidentiality of information, irrespective of its storage location. This encompasses data stored in the cloud, on devices, within physical files, and extends to intellectual property. On the other hand, cybersecurity is a specialized field within information security, that consists of employing technologies, procedures, and controls to safeguard against cyberattacks targeting systems, networks, programs, devices, and data. It encompasses the protection of all entities connected online [70].
The primary objective of cybersecurity implementation is to ensure robust security measures for

computers, servers, networks, mobile devices, and the associated data. This encompasses shielding web users from malevolent actors seeking to exploit vulnerabilities. Cyber-attacks may be orchestrated to extort businesses and users or to illicitly acquire sensitive data for nefarious purposes. Thus, cybersecurity holds paramount importance for both public and private sector entities, particularly those operating in critical industries such as healthcare or finance [113].
Although cybersecurity encompasses elements associated with safeguarding data and information systems, online privacy is specifically focused on protecting individuals' personal information and data privacy when interacting online. Data protection involves active security practices. It requires tools and procedures to protect data against unauthorized electronic access, modifications, accidental disclosure, disruptions, and destruction. This involves using physical and logical strategies to safeguard information against data breaches, cyberattacks, and accidental or intentional data loss [87]. Therefore, cybersecurity and online privacy are distinct but complementary fields within the broader realm of digital security and privacy protection [113].

2.7.1 Principles of IT Security

IT security generally relies on five basic principles which are [28]:

2.7.1.1 Integrity

Involves determining whether data has been altered during communication (either accidentally or intentionally).

2.7.1.2 Confidentiality

Ensuring that exchanged resources are accessible only to authorized individuals; also involves making information unintelligible to anyone other than the parties involved in the transaction.

2.7.1.3 Availability

Ensuring the proper functioning of the information system; The objective of availability is to ensure the accessibility of a resource or a service.

2.7.1.4 Authentication

Ensuring that resources are accessible only to authorized individuals. Authentication involves verifying the identity of a user, i.e., ensuring that each party knows that its partner is who they claim to be. Access control can allow access to resources only to authorized individuals.

2.7.1.5 Non-repudiation

Ensuring that a transaction cannot be denied; Non-repudiation of information ensures that all parties involved cannot refute the transaction.

2.7.2 Concepts of Cyber security and Cyber crime

Cybersecurity refers to the protection of individuals, ideas, and data in cyberspace. Enhancing the security of computer hardware and software would therefore increase information security [28]. Two types of individuals drive these concepts [28]:

2.7.2.1 Black hat hacker

Hackers infiltrate computer systems stealthily or covertly to steal sensitive information or create denial-of-service attacks. These kinds of individuals engage in what is called cybercrime.

2.7.2.2 White-hat hacker

White-hat hackers, also referred to as "ethical hackers," are cybersecurity experts who utilize their abilities for ethical purposes and to uphold justice. A white-hat hacker who discovers a security flaw in an application will report it to the developer, thus allowing them to improve the security of the application before it is compromised. These kinds of individuals engage in what is called cybersecurity.

2.8 Attacks on cyberspace (Cybercrimes techniques)

2.8.1 Cybercrime Definition

Cybercrime is defined as a crime where a computer is either the target or is used as a tool to commit the crime. Cybercriminals can target individuals, businesses, and enterprises, as well as national networks and industrial infrastructures, by using computers to steal their victims' private information or take control of their computers and electronic equipment [30], [39]. Cybercrimes typically fall into one of two categories [39]:

- **Computers as a target:** Every method that could endanger computer equipment is included in this category of cybercrime, such as malware and denial-of-service attacks.
- **Use of computers:** All other criminal activities that involve the use of computers fall under this category. Cyber harassment, phishing, and fraud or identity theft fall under this category.

2.8.2 *Types of cyberattaques*

The number and types of cyber attacks continue to grow day after day, in this subsection, we cannot cover all the types of attacks existing in cyberspace. In this section, we try to present the most common cyberattack techniques as depicted in Figure 2.2 [39], [123]:

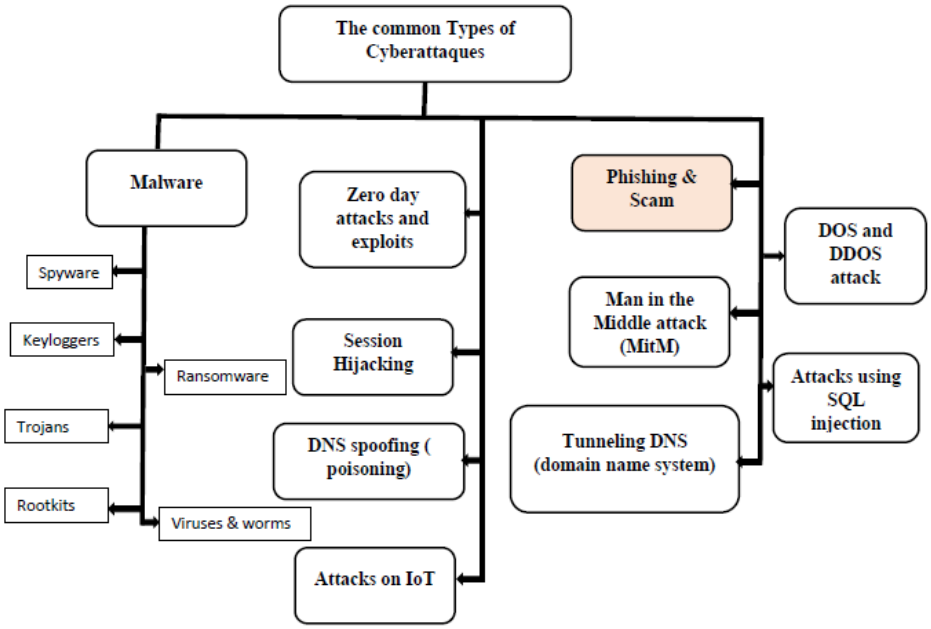


Figure 2.2: The most common cyberattacks.

2.8.2.1 *Malware*

One of the most widely used cyberattacks is malware. Malware is malicious software designed to disrupt computer networks or servers or to steal data from them. Users are tricked into downloading malware onto their devices by hackers. Once a malicious script is installed, it bypasses security measures and operates in the background, granting hackers access to sensitive information and potentially allowing them to take control. The most common malware attacks are:

- **Spyware:** This type of malware spies on user activities and sends information back to the hacker, as the name suggests. This might involve passwords, logins, and bank information.

- **Keyloggers:** Keyloggers, akin to spyware, enable hackers to capture everything typed by the user (including the corresponding website) and potentially exploit it for identity theft or blackmail purposes.

- **Ransomware:** Ransomware attacks represent a highly common type of online criminal activity. This malware encrypts users' data stored on their systems, effectively locking them out, and demands a ransom for its release.

- **Trojans:** Trojans, a category of malware, conceal themselves within trustworthy software, deriving their name from the famous Trojan horse. For example, downloading what appears to be antivirus software could result in device infection.

- **Viruses & worms:** The virus is activated when opening a program or a file that has a virus attached to it. Once activated, a virus can clandestinely duplicate itself, causing the device to operate sluggishly or leading to data deletion. Worms, categorized as viruses, spread from one compromised computer to another within the network, granting hackers remote control over the entire system.

- **Rootkits:** Malware called rootkits allows hackers to take control of the target system and access it at the administrator level. Rootkits are extremely dangerous and difficult to find because they hide deep within the operating system of a device. using a rootkit hackers can theft private data, antivirus software removal, and install keyloggers. For illustration, Kaspersky discovered a rootkit in July 2022 that could continue to operate on a victim's computer even after reinstallation or reboot.

2.8.2.2 Phishing & Scam

Phishing represents a social engineering hazard wherein users are tricked through fraudulent emails and messages aimed at acquiring their sensitive data or installing malicious software on the target system.

2.8.2.3 Man in the Middle Attack

A Man in the middle attack (MitM) attack happens when hackers compromise a network or intercept data to eavesdrop. These attacks are more frequent when using public Wi-Fi networks because they are so simple to hack. For illustration, when checking the balance in a bank account while using a cafeteria's Wi-Fi, hackers can access data and steal usernames and passwords, draining accounts later. In addition, conversations can be spoofed using MitM attacks. Hackers interject themselves into dialogues and pose as the other party believed to be spoken to.

2.8.2.4 DoS and DDoS attacks

Attacks of this kind are designed to overwhelm servers and shut down services. A DoS attack happens when cybercriminals use erroneous requests and traffic to saturate a system and bring it to

a complete stop. The same type of attack, known as a DDoS (distributed denial of service) attack, is carried out concurrently by multiple compromised devices. These cyber assaults generally target disrupting or halting business operations rather than pilfering data. Sites like Twitter, SoundCloud, and Spotify have been taken down by DDoS attacks.

2.8.2.5 Attacks using SQL injection

Most websites store confidential information such as passwords, logins, and account details in SQL databases. Hackers trickle the database into disclosing this information using an SQL injection attack. Although a little technical, these attacks involve a hacker entering predefined SQL commands into a data entry field such as login information. These commands possess the capability to access confidential data, modify database records, or initiate critical operations such as system shutdown.

2.8.2.6 Tunneling DNS

Hackers use DNS tunneling, a type of cyberattack, to get around more established security measures like firewalls and access systems and networks. Hackers hide malicious software in DNS queries and responses, which are typically ignored by security software. Once inside, the program latches onto the target server and grants the attackers remote access. Attacks using DNS tunneling are particularly dangerous because they frequently go unnoticed for a long period during which hackers can install malware, change code, add new access points, and steal sensitive data. In one instance, cybercriminals attacked Air India and other airlines using DNS tunneling to steal passport information and credit card details.

2.8.2.7 Zero-day attacks and exploits

Cybersecurity flaws known as zero-day exploits can be found in software or networks without the constructor's knowledge. When hackers gain access to a system using those flaws to steal data or cause harm, it is known as a zero-day attack. On the website phishing attack the "zero-day attack" is the new malicious website that is not yet blacklisted, that is to say, the attack is carried out before the phishing blacklist detector is updated. For instance, Apple might unintentionally include a way for hackers to steal your iCloud data in a new release of iPhone Operating System (iOS). The attacked company has "zero days" to fix the flaw after they become aware of it because they are already exposed. Microsoft, Google, and Apple all had to patch zero-day bugs in the first few months of 2022.

2.8.2.8 Attacks on Internet of Things

Cyberattacks may target IoT gadgets like smart speakers, TVs, toys, and smartwatches. The theft of data from a device or the connection of numerous IoT devices to a botnet that can launch DDoS

attacks are examples of IoT attacks. IoT devices frequently lack antivirus software, making them prime targets for cybercriminals. The largest DDoS attacks in history frequently made use of IoT devices composing bot armies.

2.8.2.9 DNS spoofing (poisoning)

Hackers can direct online traffic to a fake website by using the DNS spoofing technique. These websites resemble the victim's final destination, and any information provided therein is immediately sent to the hackers, giving them access to the victim's accounts. Such as a social media profile or a bank login page. Cybercriminals can also harm businesses by sending customers to a subpar website with bad content using DNS spoofing.

2.8.2.10 Session hijacking

Is a sort of man-in-the-middle attack, the assailant seizes control of a session between a client and a server. Without requiring any kind of authentication, the attacker's computer switches its IP address for the client's address and keeps connecting to the server. Hackers have complete control over a session once they've taken over the client's account. Consider accessing the internal database of a business while on a business trip. A hacker can access all the company's files by hijacking the session.

2.9 Cybersecurity systems and tools

At the same time as cybercrimes evolve, information protection solutions also advance. It is equally important to implement preventive measures and apply solutions and procedures in case of security breach [87]. In this section, we introduce the main technologies, tools, and systems used to ensure cybersecurity and fight against cyber threats. The anti-phishing approach is not explained in this section as it will be detailed in the next chapter. Figure 2.3 depicts the most common cybersecurity systems.

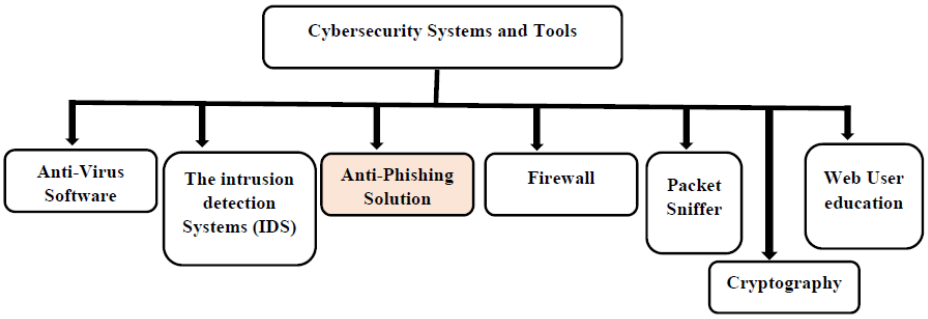


Figure 2.3: The most common cybersecurity systems and tools.

2.9.1 *Anti-virus software*

Anti-virus software is a sort of computer program designed to detect, prevent, and remove malicious software, commonly referred to as malware. The primary function of anti-virus software is to scan files, programs, and the overall system to identify any malicious code or suspicious activity. When a threat is detected, the software can either quarantine or delete the infected files or in some cases, attempt to repair them. Some antivirus software examples include Microsoft Defender, Malwarebytes, and McAfee Total Protection [51].

2.9.2 *Intrusion Detection Systems*

Intrusion Detection Systems (IDS) are software applications specifically developed to identify unauthorized access attempts or malicious behavior within a network. Some instances of IDS include Suricata, OpenWIPS-NG, and SolarWinds Security Event Manager [51]. Figure 2.4 presents a basic architecture of an IDS [81].

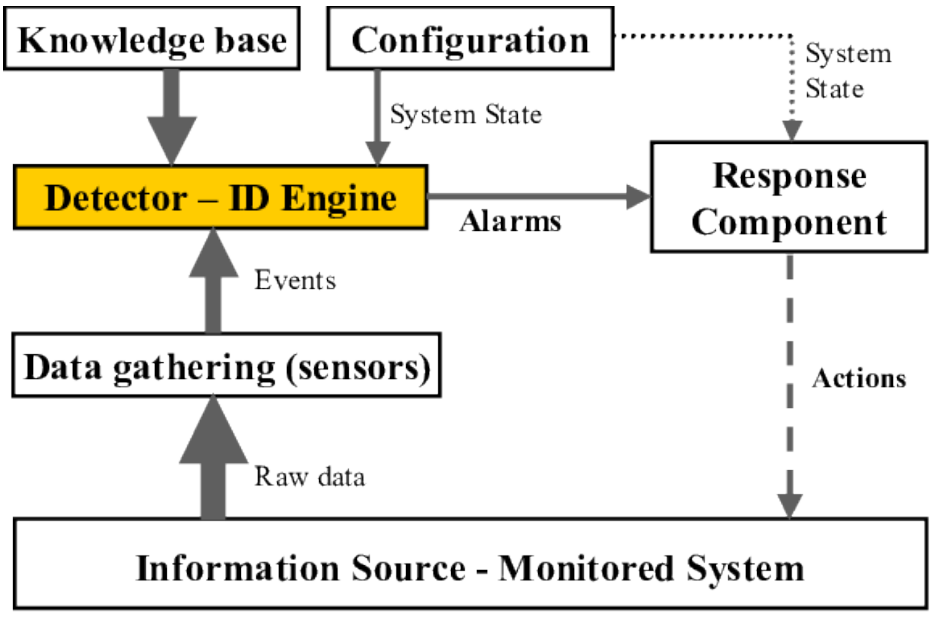


Figure 2.4: A general architecture of an intrusion detection system (IDS) [81].

2.9.3 *Anti-phishing solutions*

(This subsection will be discussed in detail in the next chapter)

2.9.4 *Firewall*

A firewall is a network security tool that monitors the flow of network traffic, both incoming and outgoing, and decides whether to allow or deny data packets based on a predefined set of security rules. Its main objective is to create a protective safeguarding barrier separating the internal network from external entities, effectively blocking malicious traffic such as viruses and hackers from infiltrating. [53]. Figure 2.5 presents a layered firewall architecture [90].

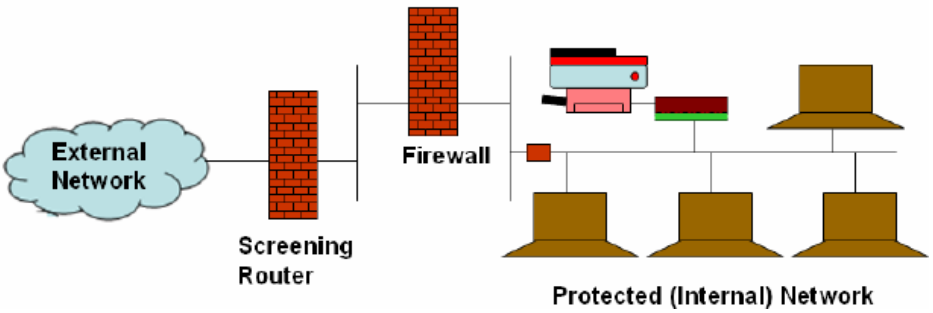


Figure 2.5: An architecture of firewall [90].

2.9.5 *Packet sniffer*

Packet sniffers, whether in the form of physical devices or software applications, are utilized to identify, capture, and store data packets transmitted across a network. When employed by network administrators, these tools actively observe internet traffic, enabling real-time monitoring of data, which can subsequently be analyzed to assess and troubleshoot performance issues across servers, networks, hubs, and applications [21]. Figure 2.6 displays the functioning of a packet sniffer [137].

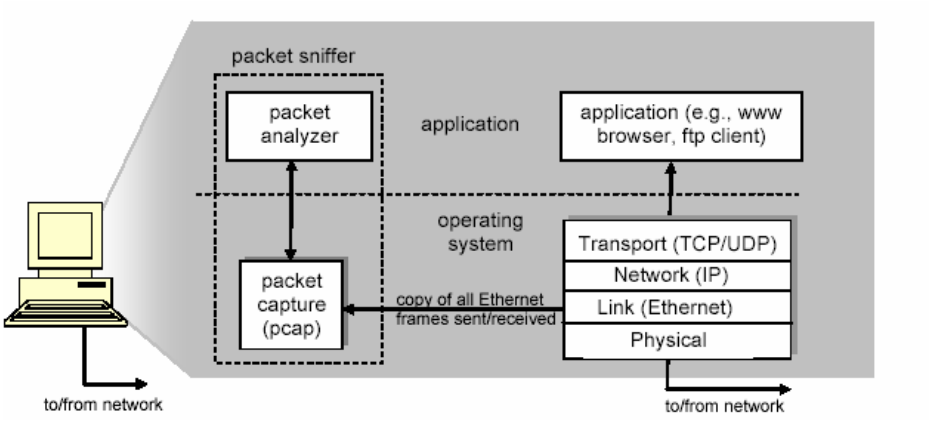


Figure 2.6: An architecture of the packet sniffer [137].

2.9.6 Cryptography

It is a discipline of cryptology that focuses on protecting messages using keys. Cryptography aims to make the message unintelligible to anyone other than the intended recipient. It thereby ensures the confidentiality of exchanges. It typically relies on various cryptographic algorithms such as Data Encryption Standard (DES), Rivest-Shamir-Adleman (RSA), etc [28].

2.9.7 Web user education

Web user education tackles the most uncertain aspect of cybersecurity: Any individual can inadvertently introduce a virus into an otherwise protected system by neglecting to adhere to proper security protocols. but to avoid social engineering attacks that aim to exploit the naivety of users, it is crucial to impart knowledge to users regarding practices such as deleting suspicious email attachments, not clicking on suspicious links or feeling sensitive information to an uncertain source, refraining from connecting unknown USB drives, and other essential lessons, as this significantly contributes to the security of any organization [72].

2.10 Conclusion

In this chapter, we have presented an overview of online privacy. In particular, we introduced the definition and the impact of the COVID-19 pandemic on the protection of privacy on the internet, an overview of the Algerian legislation that frames and regularizes this area, then we mentioned the various principles and threats to online privacy. In addition, we focused on the subject of digital security, by introducing the definition of information security, online privacy, cybersecurity, and cybercrime. Moreover, we tried to mention the most common types of cyberattacks such as malware, phishing, scams, a man in the middle, DoS attacks...etc. And finally, some systems and tools to fight against these attacks note that no protection solution is perfect, furthermore, this field remains vast and complex and far from being encompassed in its entirety within the framework of a single chapter. The study we conducted in this chapter has enabled us to understand the challenges and threats to online privacy to better grasp, in the following chapter, the solutions, and mechanisms to address this field and narrow down the security domain to which we will contribute, which is the domain of phishing websites detection.

Chapter. 3

Phishing: state of the art

3.1 Introduction

In recent times, individuals have developed a stronger reliance on the Internet, particularly with the advent of the COVID-19 pandemic, where almost all activities have shifted online, such as bill payments, banking transactions, online shopping, and e-commerce. People's lives have become more convenient. However, this digital transformation has also brought about privacy concerns for internet users. The vast amount of sensitive data circulating online has attracted cybercriminals who exploit this opportunity to execute various forms of social engineering cyberattacks, such as phishing. Phishing is a form of social engineering cyberattack, where deceptive websites are employed to deceive individuals into divulging sensitive information to the attacker. This information may include personal account details, credit card information, bank account credentials, login identifiers, and other confidential data. Lately, there has been a significant increase in phishing attacks, and the security challenges we encounter as a result of phishing are growing rapidly. In 2022, phishing reached an all-time high, as confirmed by the APWG, which documented over 4.7 million attacks. Since the start of 2019, the frequency of phishing attacks has been increasing at a rate of more than 150% annually. All these statistics affirm that phishing attacks remain a crucial threat to cyberspace security and a source of interest for researchers in this domain. Various methods to combat phishing have been explored in the literature, including list-based, visual similarity, heuristics-based, and machine-learning approaches, among others. Each approach comes with its own strengths and weaknesses. For instance, the blacklist approach falls short in detecting non-blacklisted phishing sites (referred to as zero-day attacks), while visual similarity techniques relying on image analysis can be computationally intensive and resource-consuming. In recent times, deep learning techniques have gained significant traction in the field of phishing detection. This surge in popularity is fueled by the rapid advancements in deep learning and the subsequent success achieved through numerous studies conducted in this area. In this chapter, we present a literature review of the phishing field, defining all the basic concepts such as the definition of the

phishing attack, phishing motives and statistics, the types of attacks, the different anti-phishing solutions, and some related works to better understand this area.
This chapter was the subject of our two conference papers "Survey paper: Taxonomy of website anti-phishing solutions," (SNAMS2020) IEEE, Paris, France [105]. and "A literature survey on anti-phishing in websites" (NISS2021), ACM, KENITRA, Morocco.[106]

3.2 Definition

Initially, telephone networks were targeted by phishing attacks, commonly referred to as Phone Phreaking from the 1950s till the 1980s. This led to the term "fishing" being replaced with "phishing" in English, as it shared the same concept. The term "phishing" was introduced by a group of hackers in 1996 when they exploited the Internet to steal America Online (AOL) accounts. Their method involved deceiving unsuspecting AOL users into disclosing their passwords [135], [18]. Figure 3.1 presents the progression of phishing attacks from 1996 to 2020 [44].
Phishing is a form of social engineering attack that seeks to manipulate individuals into divulging confidential information. To execute such an attack, hackers must replicate legitimate websites and subsequently distribute the links of these malicious sites via email, instant messages, social networking platforms, multiplayer games, and various other channels. Once victims access these deceitful links, they unknowingly provide their sensitive data under the pretense of interacting with legitimate websites. Consequently, the attacker gains access to personal information such as usernames, login credentials, passwords, financial account details, social media data, and personal addresses. These private credentials are frequently exploited for malicious purposes, including but not limited to identity theft, financial gain, seeking notoriety, damaging reputations, and other various illicit activities [135], [104]. Figure 3.2 presents the steps of performing a phishing attack.

3.3 Phishing Motives

The main motivations behind carrying out the phishing attack are [98], [58]:

- Financial Gain: The theft of banking credentials, such as credit card identifiers, might yield monetary benefits for the attackers.
- Identity Theft: Phishers use sensitive user data to steal identities and engage in activities by pretending to be the victim.
- Unsavory Reputation: Phishers sometimes commit fraud to gain fame among hackers.
- The theft of trade secrets and documents: Attackers use spear phishing to steal sensitive data from specific companies, generally for competitive reasons.

Figure 3.1: The progression of the phishing attack from 1996 to 2020 [44].

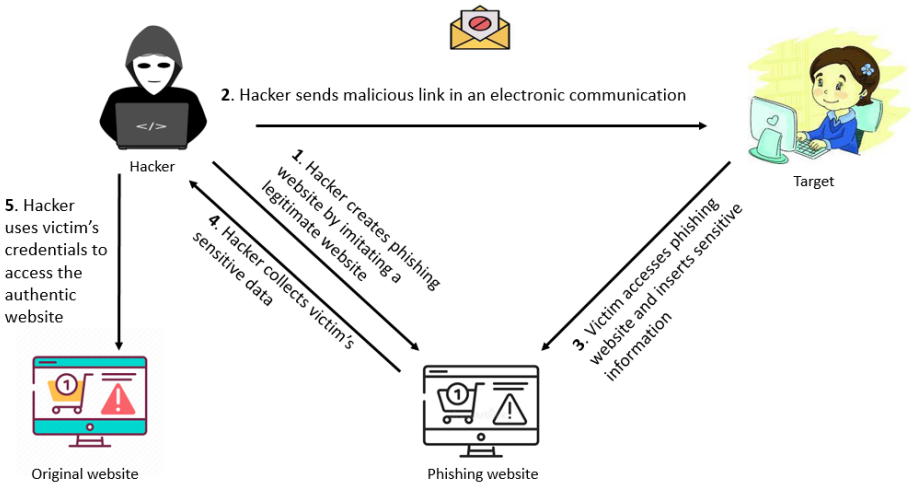


Figure 3.2: The stages of the phishing attack.

- Ethical hacking: involves identifying flaws and vulnerabilities within a system or emerging technology to wring security gaps.

3.4 Phishing Statistics

Nowadays, phishing has become a major concern as the number of phishing attacks is increasing every year. Based on the APWG's report for the fourth quarter of 2022, it was noted that 2022 set a new high-water mark for phishing incidents, as the APWG recorded over 4.7 million such attacks. In comparison to the early part of 2019, the rate of phishing attacks has been increasing at an annual growth rate exceeding 150%. During the last quarter of 2022, the APWG documented 1,350,037 phishing attacks, marking a slight increase from the previous quarter when they recorded 1,270,883 such attacks. Notably, the third quarter had already set a new record as the worst period for phishing observed by the APWG. Additionally, in the fourth quarter of 2022, the APWG saw a rise in phishing email submissions from both its members and the public. In October 2022, they received 101,104 distinct email subjects, the highest number ever recorded by the APWG [14]. Over four years, the APWG has witnessed substantial growth, with the rate of increase surging to over 150% annually, as illustrated in Figure 3.3.

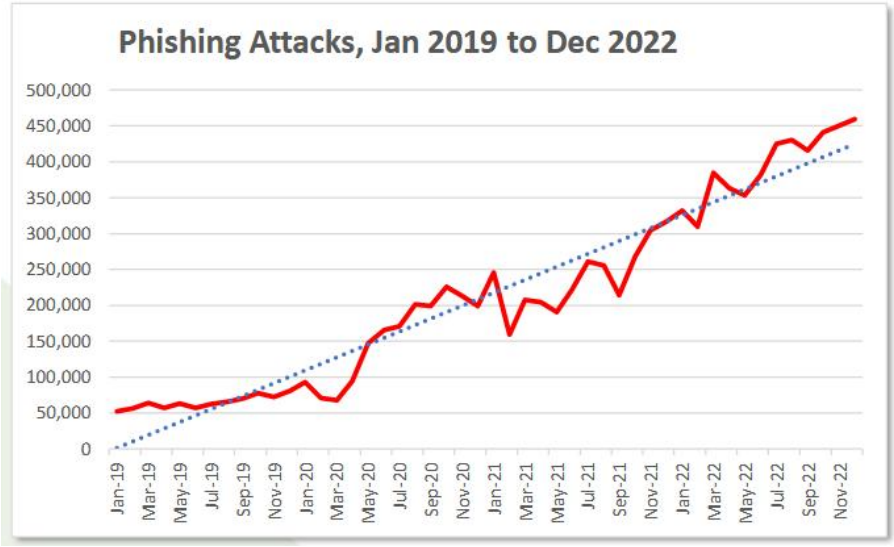


Figure 3.3: Phishing attack statistics from 2019 to 2022 [14].

Figure 3.4 illustrates the industry sectors that were the primary targets of phishing attacks, based on data from the APWG's report for the fourth quarter of 2022. As shown in the figure, the largest share of phishing attacks, constituting 27.7 percent, targeted the financial sector, which includes banks. This increase can be attributed to the decline in physical store sales resulting from the onset of the COVID-19 pandemic. Following closely were attacks against webmail and software-as-a-service (SAAS) providers, making up 17.7 percent, with a slight drop from the previous quarter.

Approximately 6 percent of attacks were directed at payment processors such as PayPal, Venmo, and VISA. Phishing attacks against social media companies exhibited a declining trend after fluctuating from 8.5 percent of all attacks in the fourth quarter of 2021 to 15.5 percent in the second quarter of 2022. Phishing incidents targeting cryptocurrency-related entities, such as cryptocurrency exchanges and wallet providers, decreased from 4.5 percent in the second quarter to 2.0 percent in the third quarter and 2.3 percent in the fourth quarter, reflecting the cryptocurrency market's volatility with falling values [14].

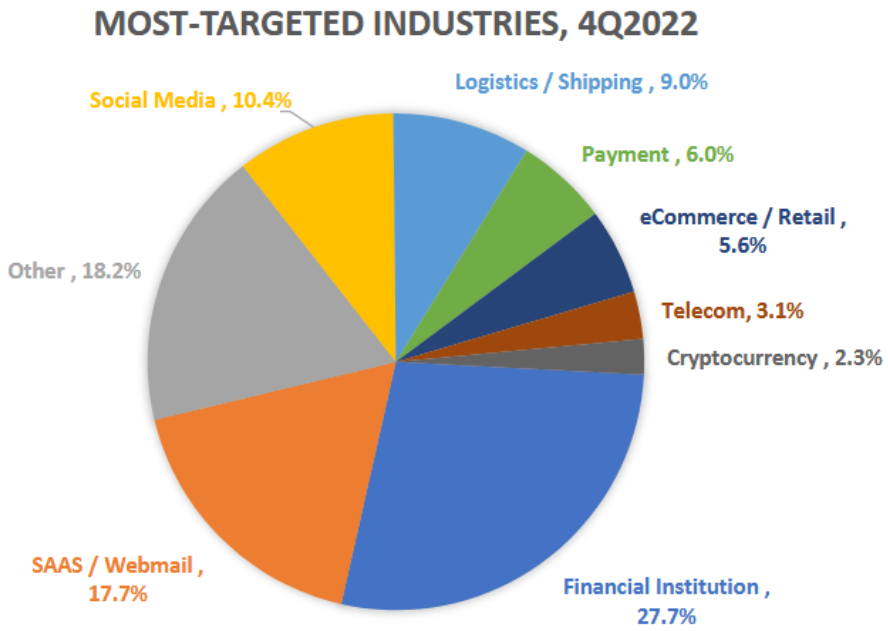


Figure 3.4: The Most Targeted Industries by Phishing According to the APWG 4Q 2022 Report [14].

3.5 Types of phishing attacks

Phishing is among the prevalent and most common cyberattacks, it comes in various forms and is categorized based on the tactics employed by the attacker to acquire the personal information of the victim. The phisher can employ deceptive methods or resort to technical subterfuge, unbeknownst to the users, to execute a phishing attack [105]. Phishing attacks can be categorized into two main classes namely malware-based phishing and deception-based phishing and each class relies on various sub-methods [19], [98], [105]. Figure 3.5 presents a taxonomy of the phishing attack types.

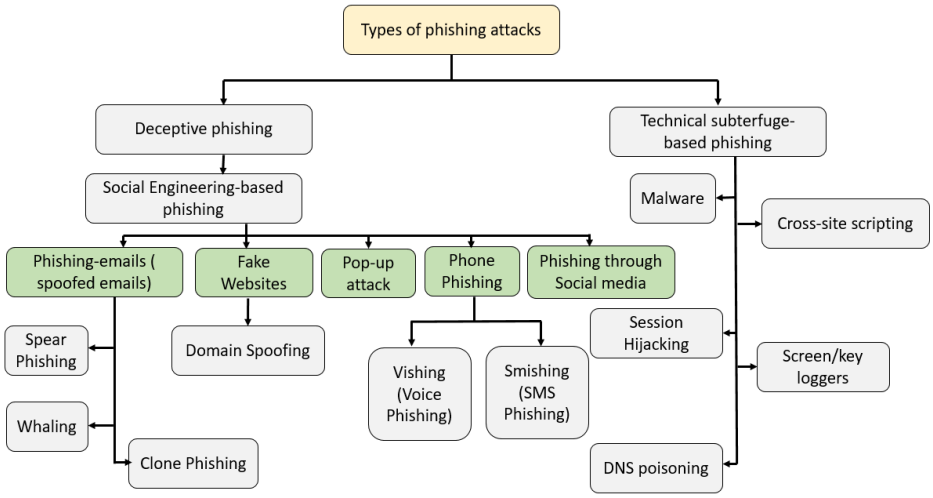


Figure 3.5: Taxonomy of phishing attack types.

3.5.1 Deceptive phishing

In deceptive-based phishing, a hacker sends deceptive emails that appear to come from a trusted institution. The phisher prompts users to click on a link that redirects them to a fraudulent website, aiming to trick them into revealing private information. The phisher then exploits this information for illegitimate purposes [19]. This type of attack is based on the use of deceptive techniques for phishers falsely representing themselves as a legitimate company to deceive their targets into believing they are under a cyberattack. Subsequently, users unknowingly click on a malicious link, leading to their computers being infected with malware [54].

3.5.1.1 Social engineering

Social engineering refers to the manipulation of victims through psychological tactics to deceive them into divulging personal and valuable information or granting unauthorized access. It involves

the pressure exerted on individuals to disclose sensitive information, while also encompassing the creation of convincing scenarios and contexts by phishers and other malicious actors [98], [49], [54].

1. Phishing emails (Spoofed emails)
   In email phishing, the phisher sends an email that appears legitimate and is intended to fool the recipients by impersonating a trusted company that the victims deal with, such as an online retailer or a bank. By tricking the recipients into clicking on a malicious link to reset their login information or account password, the hacker can obtain their sensitive data [54]. Figure 3.6 depicts an example of phishing email [66].

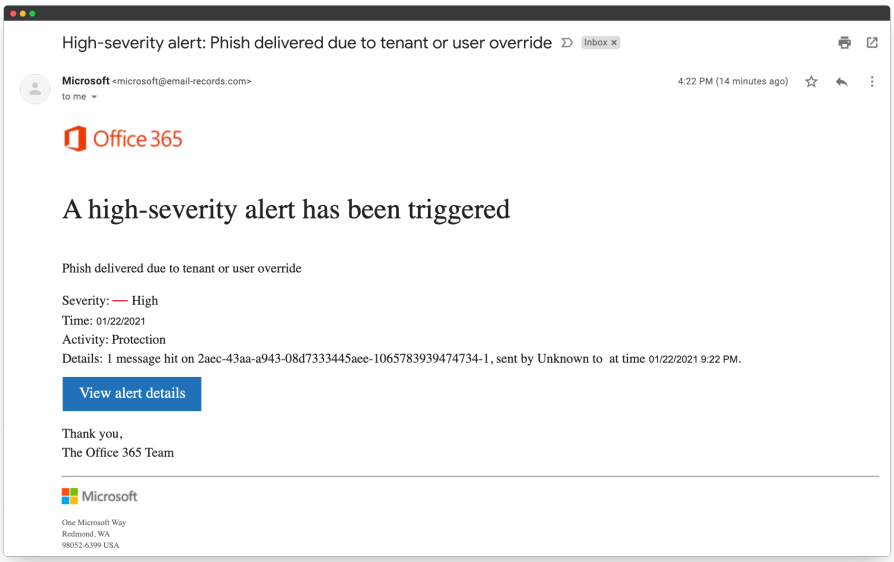


Figure 3.6: Example of a phishing mail [66].

There are several types of attacks that belong to the phishing email category, such as:

- Spear phishing
  In this type of attack, phishers target a small group or specific individuals within an organization, they gather all the information they require about their targets from social media and other websites, including their name, position, and contact information, and use it to defraud and manipulate their targets. This type of phishing is the most frequently used to breach companies' defenses. Attackers who use spear phishing frequently gather and use specific statistics about their targets to improve their chances of success [98], [54].

- Whaling

  A whaling attack is a type of phishing that targets senior executives and other notable targets, including top government officials, CEOs of large corporations, and major business administrators. Phishers devote a lot of time and resources to finding these victims, searching for any opening, and then patiently waiting for the ideal opportunity to strike. These attacks are very challenging to track down because they rely solely on social engineering (typically deceptive email) and don't use any malware or bogus websites. A successful whaling attack can gain access to important data because the target people frequently have extensive access to delicate parts of the network [98], [54], [58].

- Clone phishing

  In a clone phishing attack, a hacker clones an original email that the target has already received. In addition, the attacker uses address spoofing to make it appear to be coming from the original sender. The attacker then tricks the victim into thinking that the email is being resent or updated by replacing the legitimate link in the email content with a malicious link [98], [54].

2. Fake website (website spoofing)

   A fake website is a commonly employed phishing attack, where a hacker creates a fake website that closely resembles a legitimate one, tricking users into believing it is the genuine site that they usually access. When victims unsuspectingly log in to their accounts on these fake websites, their login credentials are captured by the attacker. Previously, attackers used web spoofing with domain names that look different than the legitimate domain, making it easier for users to identify the fraud. However, now phishers combine domain spoofing with web spoofing, creating websites that are more convincing in luring victims. Generally, the links to fake websites are embedded with phishing emails or advertisements. Additionally, these fraudulent websites may rely on users mistakenly selecting the wrong site in their web browser, leading them directly to the deceptive site. For example, the Amazon website has been spoofed, by mimicking the authentic one but with a different URL. The hackers replicated the fonts, images, and other details to create a convincing facade, and they were relying on users entering their login credentials, hoping to acquire sensitive information [54], [58].

   - Domain Spoofing

     To trick people into thinking it is a legitimate website, the phishers use a modified domain that looks familiar, like "faceboook.com" or "amazoun.com," and it may also contain many subdomains, like "paypal.secure.transaction.com." This trick misleads users who click links to get urgent information or users who didn't even know what the website looks like [54]. Figure 3.7 depicts an example of this attack where the attacker imitates a Facebook page and replaces the domain name with "faceb00k" instead, exploiting visual similarity to deceive users [76].

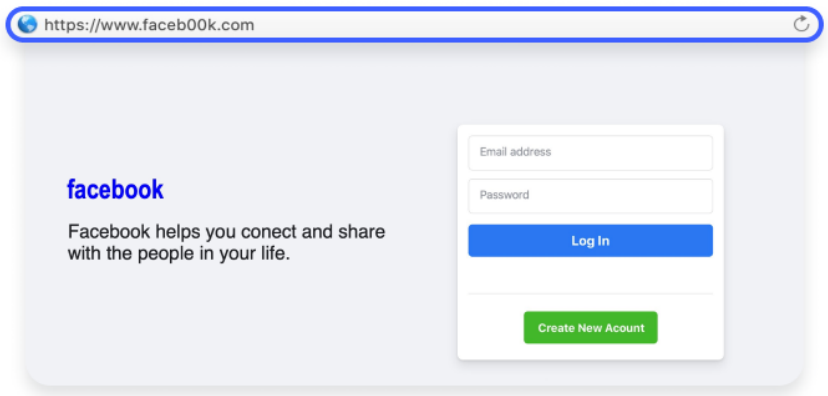


Figure 3.7: An example of domain spoofing attack [76].

3. Pop-up attack
   This type of attack consists of displaying a deceptive pop-up window on legitimate websites, tricking users into thinking that the pop-up originates from the adjacent legitimate site they are visiting. The purpose of this attack is to deceive users into divulging sensitive information or engaging in harmful activities, including clicking on malicious links or downloading files infected with malware. These pop-up windows can claim to be, fake software updates, lottery-winning notifications, warnings about system errors, or fake safety alerts to trick users into complying with their requests [19], [54].

4. Phone phishing
   Hackers deceive people over the phone using social engineering by posing as a reputable authority. This technique is divided into two sub-techniques, which are:

   - Vishing (voice phishing)
     The term "voice phishing," also known as "vishing," refers to the practice of the attacker who attempts to obtain sensitive information from victims over the phone (executing a phone call). The caller might pretend as a trustworthy person [54].
   - Smishing (SMS Phishing)
     Phishing via text message or Short Message Service (SMS) is known as "smishing." The attacker selects his victims and searches for their phone numbers. Once he has them, he sends the victims an SMS that contains a phishing URL and instructs them to click it and enter their credentials in the pop-up form to reset their account information [54].

5. Phishing through Social Media
   Social networking platforms like Facebook, Twitter, and Instagram have become prime targets for phishing attacks. These sites facilitate communication and information sharing among users, providing phishers with ample opportunities to engage in fraudulent activities. By impersonating familiar individuals on these platforms, phishers exploit users' trust to obtain sensitive data, ultimately seeking financial gain through the popularity of these sites [44].

3.5.2 Technical subterfuge

Technical subterfuge is a technique for fraud in which the hacker relies on technical tricks like malicious code embedded in emails or websites, self-executing code, and malicious software installed on PCs to steal sensitive data directly [58]. There are several types of phishing based on technical subterfuge such as:

3.5.2.1 Cross-site scripting

This attack is also known as content poisoning. It takes place when an attacker inserts malicious JavaScript code into a legitimate page that is viewed on the victim's site. While surfing the webpage and clicking a random button, the victims are then redirected to a phishing URL. After that, the attacker can steal the login information and abuse it [58].

3.5.2.2 Malware

Phishers install malware on their victims' computers. This malware intercepts requests from particular websites and sends users to the corresponding phishing website in order to steal sensitive information. This attack is very dangerous because it leads victims blindly to the trap [58].

3.5.2.3 Session Hijacking

In Wireless Local Area Network (WLAN), session hijacking is a frequent and serious threat. Another name for this is cookie hijacking. In this case, the session key is taken via DoS attack to steal the user's identity and gain unauthorized access to the resources. The attacker tries to disconnect a specific access point from the mobile station [58].

3.5.2.4 Screen loggers and key loggers

This is very difficult to track down, and screen loggers no longer work with virtual keyboards. They send screenshots and mouse movements to the attacker who is located far away [105].

3.5.2.5 DNS poisoning

This attack is one of the riskiest phishing methods. The phisher does not employ regular tactics such as tricking victims into clicking phishing URLs, requesting their credentials, or sending spoofed or cloned emails, etc. Even when they search for or enter a legitimate URL, victims have been automatically redirected to a phishing website, this is accomplished by altering DNS entries or downloading malicious software onto the computers of victims. Once the victim establishes a connection with the fake DNS server, they are redirected to malicious websites or have malware installed on their computers [58].

3.6 Anti-phishing Solutions

Numerous strategies have been devised to combat phishing attacks, falling into two main categories: software-based solutions and efforts to raise user awareness. Figure 3.8 depicts a detailed hierarchy of these anti-phishing methods, offering a clearer insight into their classification.

3.6.1 Enhancing user consciousness

Phishing attackers consistently exploit inexperienced individuals to achieve their goals. Therefore, it is crucial to educate end-users so that they gain a deeper understanding of the characteristics of phishing attacks, enabling them to accurately distinguish between phishing and legitimate messages. This approach is known as enhancing user awareness [15], [75].

3.6.2 Software based-solutions

It encompasses specialized software designed to automatically classify phishing and legitimate messages on behalf of users. Their primary goal is to address the gap resulting from human errors or ignorance. This is crucial, as user training is a more costly endeavor compared to implementing automated software classifiers. These solutions can be categorized into two main groups: content-based approaches and non-content-based approaches [15], [75].

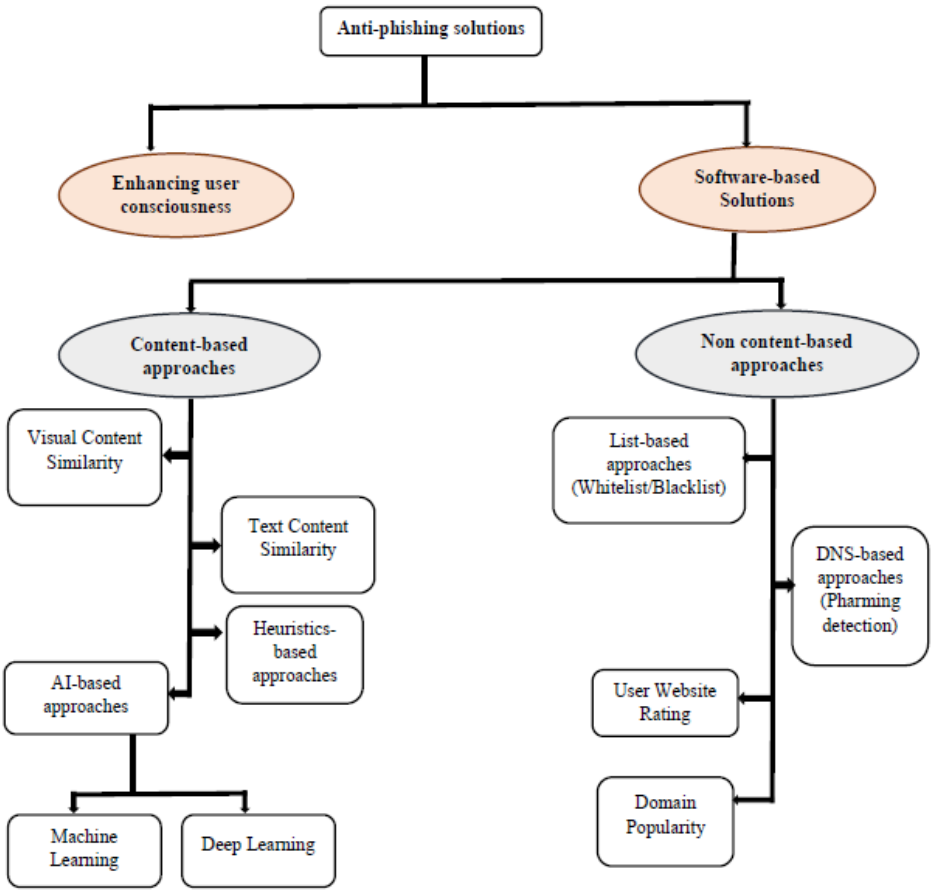


Figure 3.8: A hierarchy of anti-phishing approaches.

3.6.2.1 Content-based anti-phishing approaches

This technique relies on the analysis of website content to detect phishing attacks. This analysis involves examining various elements, including URL features, images, and text content, such as password fields, spelling and grammar checks, and assessing visual similarities between websites [105], [106]. This type includes several approaches such as:

- Visual content similarity
  This approach involves assessing the visual aspects of web pages, which includes comparing elements like images, design styles, logos, and more on suspicious websites, with a legitimate database to determine a similarity score. If the similarity score exceeds a certain threshold, the website is classified as a phishing site; otherwise, it is considered legitimate. One drawback of this method is that it demands a significant amount of time for comparisons and consumes substantial storage space for the dataset, potentially leading to higher false negative rates. Additionally, the visual features blacklists or whitelists require frequent

updates. However, this approach is effective in addressing zero-day phishing attacks, which URL blacklists cannot do.

- Text content similarity
  Phishers often employ keywords that closely resemble real terms, for example on deceptive social networking sites they use "Facebouk" instead of "Facebook," in an attempt to deceive online users and trick them into revealing their login information. To counter this kind of fraud, a text content similarity-based method is utilized to analyze textual elements, including HTML codes, keywords, scripts, and so on. This approach involves maintaining a keyword list for verification purposes. As an illustration, researchers introduced a technique outlined in [46] called GoldPhish, which relies on visual content similarity to identify phishing websites. Initially, the method captures a screenshot of the webpage and employs optical character recognition (OCR) to extract textual elements, including those within a logo. Subsequently, the extracted text is inputted into the Google search engine for analysis of the search results. This evaluation involves verifying if the domain name of the relevant website matches the queried website. A discrepancy indicates a disparity between the portrayed identity (relevant website) and the actual identity (queried website), leading to the determination that it is a phishing webpage.

- Heuristics-based approaches
  This method relies on certain heuristics related to phishing and legitimate websites or emails, including elements like URLs, web text content, email text content, source code, and more. However, it often uses URL analysis, considering features like IP-based URLs, the presence of phishing-related terms in the URL, and the number of dots, among others. Typically, it is coupled with artificial intelligence algorithms for binary classification. This approach is effective in identifying zero-day attacks, but as noted in [25], there is a significant risk of mistakenly categorizing legitimate websites using this technique. The CANTINA method discussed in [138] employs content analysis utilizing the Term frequency-inverse document frequency (TF-IDF) algorithm. This algorithm, commonly used in text mining and information retrieval, assesses the significance of words within a specific document among a collection of documents. It then identifies the five terms with the highest frequency to create a lexical signature of the webpage. Subsequently, this lexical signature is cross-referenced in the Google search engine to determine the legitimacy of the website.

- Artificial intelligence-based approaches
  John McCarthy, often acknowledged as a key pioneer in the field of Artificial Intelligence, characterized AI as the discipline and practice of crafting intelligent machines. Artificial Intelligence (AI) is alternatively described as a subfield within computer science focused on simulating intelligent actions in computer systems [124]. AI involves the endeavor to transfer human learning and cognitive abilities into computers, thereby imbuing them with intelligence. Rather than requiring explicit programming for every task, AI systems can

autonomously discover solutions and tackle problems. The overarching objective of AI research has consistently revolved around comprehending the workings of the human brain and mind on one side and enabling their artificial replication on the other. Figure 3.9 presents a hierarchy of artificial intelligence domain [68].

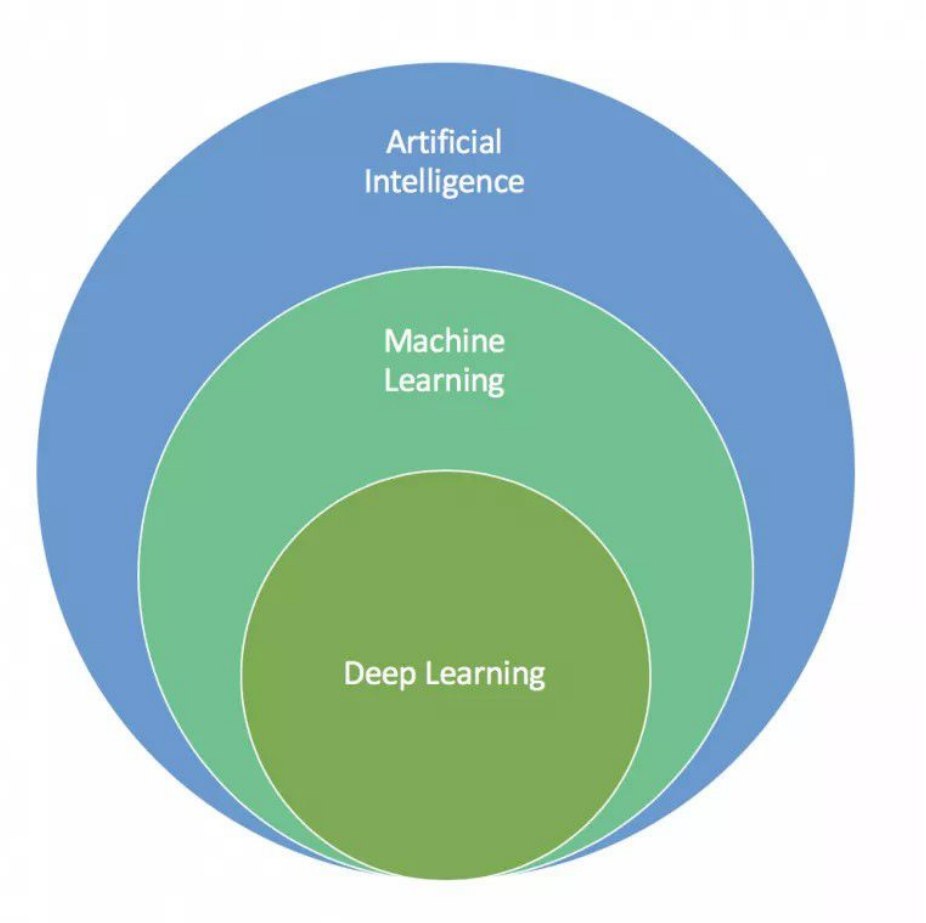


Figure 3.9: Hierarchy of AI [68].

– Machine Learning

  Machine learning, a subset of artificial intelligence, is a technological approach that enables computer systems to acquire knowledge directly from real-world instances, experiences, and data. Arthur Samuel the American pioneer of machine learning and artificial intelligence said in 1959: "Machine learning is the discipline that gives computers the ability to learn without being explicitly programmed" [103]. This empowers computers to execute intricate tasks with intelligence, as they gain proficiency by extracting insights from data rather than adhering to pre-programmed instructions. Traditional programming methods depend on fixed, predefined instructions that outline

a systematic problem-solving process. In contrast, machine learning systems are assigned a specific objective and provided with a substantial dataset to serve as instances of how to accomplish that objective or to discern patterns. These systems subsequently acquire knowledge on the most effective way to attain the desired results [124]. ML utilizes algorithms and neural network architectures to assist computer systems in improving their performance progressively. These algorithms autonomously construct a mathematical model using sample data, referred to as 'training data,' enabling them to make decisions without requiring explicit programming for those decisions [52]. It is frequently employed to identify phishing websites through classification algorithms. Similar to heuristic solutions, machine learning-based techniques can address the limitations of list-based approaches by detecting previously unseen phishing attacks (zero-day attacks). Notably, machine learning techniques possess the ability to autonomously develop their classification models by analyzing extensive datasets. Finally, The success of this method relies on various factors, such as the magnitude of the training set (larger datasets result in improved recognition capabilities), the chosen features and the classifier employed [75].

– Deep learning
  DL a branch of machine learning, employs multiple layers of algorithms organized as neural networks. Input data undergoes analysis through different network layers, where each layer identifies particular features and data patterns. The crucial element of deep learning is that these feature layers aren't manually crafted by human engineers; instead, they are derived from data using a versatile learning process to discover the attributes necessary for categorizing specific patterns [124], [100]. DL is an illustration learning approach, that gained preeminence within the field of AI in recent years. This surge in attention is attributed to its remarkable achievements across a multitude of applications. At its core, Deep Learning revolves around the concept of automatically extracting feature representations from data, removing the necessity for manual feature crafting. In the realm of phishing detection, Deep Learning has emerged as a formidable force, leveraging various classifiers, including Recurrent Neural Network (RNN), LSTM, and CNN, to surmount the limitations associated with traditional machine learning techniques, which often require manual attribute extraction. Both CNN and LSTM are widely employed in the domain of time series analysis. LSTM models demonstrate proficiency in capturing extended relationships within sequential data, while CNN models are adept at learning localized patterns within the data [82], [83], [110], [96].

3.6.2.2 Non-content-based anti-phishing approaches

- List-based approaches (blacklist/whitelist)
  A simple and widely used method relies on a list of website URLs, categorizing them as ei-

ther phishing or legitimate. This compilation of known phishing URLs is typically denoted as a blacklist, while a whitelist is reserved for trusted sites. Consequently, the classification of a URL as either phishing or legitimate is determined by the specific list it belongs to. The blacklist is often utilized by popular anti-phishing tools like Google Safe Browsing [67]. The chief advantage of this list-based approach lies in its simplicity and swift implementation. However, it comes with the limitation of necessitating frequent updates to ensure the list remains current. Additionally, the blacklist is unable to identify zero-day phishing attacks originating from URLs not yet included in the blacklist [15], [75], [105], [106], [48]. G. Sonowal et al [120] introduced PhiDMA, a multi-filter solution comprising an auto-updating whitelist filter, a filter for universal resource locator attributes, a lexical signature filter, a string matching filter, and an accessibility score comparison filter. They devised an algorithm that evaluates five chosen URL characteristics. The system enhances the whitelist by incorporating URLs that successfully pass all filters and are verified as legitimate.

- Domain Name System-based approaches (pharming detection)
  This approach relies on DNS data to ascertain the legitimacy of domain names and their associated IP addresses to detect phishing attempts. The information used for phishing identification can differ between methods, such as gathering IP addresses linked to domain names, analyzing domain query logs to pinpoint frequently visited hosts...etc [105], [106], [130]. C.S., Chen et al [31], developed a tool for extracting page signatures from online content, operating on the client side. This tool encodes the current webpage to generate its signature, which is then transmitted to a remote server via a DNS query. The server compares this signature with known phishing page signatures. Should a match occur, a policy enforcement mechanism triggers, potentially alerting the user or blocking access to the page.

- User website rating
  In user website rating, feedback about the website is gathered from the users and based on these responses, the website's trustworthiness is determined. When customers visit a website, they are prompted to assess its legitimacy, allowing the website to be categorized according to user responses. They look at other features, including this one, to decide if the site is a scam or not [29]. Crowdsourcing: The Web of Trust (Web of Trust (WOT)) is a browser extension based on crowdsourcing, relying on user ratings of visited websites [29, 140]. It shields users from threats that require human judgment, such as scams, unreliable web stores, and questionable content. WOT operates through a patented system where user behavior is continuously monitored and analyzed to determine website ratings. When users search for content in a search engine, the search results are accompanied by indicators in the corner. Green indicates a trusted site, yellow indicates doubtfulness, and red indicates suspicion.

- Domain popularity

The approach based on the popularity within the domain can be considered as a non-content heuristic approach, it decides the legitimacy of a website based on certificate details, domain registration details, certificate authorities, etc. If a user clicks on a suspicious link, the browser extension sends the link to a controlled server and extracts attributes such as domain name, validity, and certificate authorities, then the information verification is made with Google, and depending on the results, the toolbar will warn the user [29]. Restricted from filling [33, 50] serves as an anti-phishing browser extension, which monitors user credentials and alerts users when attempting to input such information into fraudulent websites. User credentials are stored and safeguarded with a master password. Upon revisiting a site, users need not re-enter their credentials; instead, they can click on the browser extension's provided icon. Once logged into the browser extension, users can subsequently log into any website without re-entering credentials. The anti-phishing tools [33, 50] maintain a database to store user login details, which can be accessed from any system by simply installing the extension or toolbar.

Table 3.1 provides a comparison between the software anti-phishing solutions.

Table 3.1: Comparison between the software anti-phishing solutions.

Anti-Phishing Solution	**Advantages**	**Limitations**	**Classic method**	**Emerging method**
List-based solution	-Simple and swift to implement	-Necessitates frequent updates to ensure the list remains current. -unable to identify zero-day phishing attacks	✓	
Visual content similarity	-Effective in addressing zero-day phishing attacks, which URL blacklists cannot do.	-Demands a significant amount of time for comparisons and Consumes substantial storage space for the dataset. -Visual features blacklists or whitelists require frequent updates. -Fails to detect when the visual appearance changes slightly leading to higher false negative rates.	✓	

Table 3.1: Comparison between the software anti-phishing solutions.

Anti-Phishing Solution	**Advantages**	**Limitations**	**Classic method**	**Emerging method**
Text Content similarity	-It can effectively target and identify suspicious content that mimics legitimate platforms or brands by focusing on similarities between known keywords and those used by phishers. - It can complement other anti-phishing techniques, providing an additional layer of defense against phishing attacks.	The keyword list needs frequent updates and maintenance which is time-consuming and resource-intensive.	✓	
Heuristics-based approach	- Effective in identifying zero-day attacks	There is a significant risk of mistakenly categorizing legitimate websites using this technique	✓	
Machine Learning	- Can detect zero-day phishing attacks. -autonomously develop their classification models by analyzing extensive datasets.	Necessitate manual features identification and extraction.	✓	
Deep Learning	-Eliminates the demanding and intricate phase of feature engineering present in machine learning. It generates new features through its processes and techniques. -it can work even in a substantial volume of data. -it has shown excellent results in several fields.	-Necessitates high-performance hardware, and advanced graphical processing units, which come with a hefty price tag and demand significant time and expertise, especially when dealing with large datasets		✓

Table 3.1: Comparison between the software anti-phishing solutions.

Anti-Phishing Solution	**Advantages**	**Limitations**	**Classic method**	**Emerging method**
DNS-based approach	-Simple to implement. -detect a zero-day phishing attack.	-False positives could occur because opting for dotted-decimal IP addresses instead of domain names might be preferable under specific circumstances. -it is difficult to distinguish between benign and malicious activity solely based on DNS data. -Sophisticated attackers may employ DNS tunneling techniques to bypass DNS-based detection mechanisms, using covert channels within DNS traffic to exfiltrate data or evade detection.	✓	
User website rating	-Provides rapid detection and response to potential phishing attempts. -Involving users in the rating process empowers them to contribute to online safety which enhances the user awareness about phishing detection.	User ratings may vary based on individual perceptions, experiences, and biases, leading to inconsistencies in website trustworthiness assessments. -cannot detect zero-day phishing attacks. -Vulnerability to manipulation by malicious actors by artificially inflating the trustworthiness of phishing websites or unfairly tarnishing legitimate websites, compromising the effectiveness of the rating system.	✓	
Domain popularity	Reduced reliance on user judgment	-Cannot detect zero-day phishing attacks. Depends on third-party services which are time-consuming and disadvantageous since the absence of these services will restrict the performance of the task.	✓	

3.7 ML and DL-based related works

Several phishing detection works have been introduced in the literature, in this subsection, we delve into the most recent related studies as well as those that utilized the same dataset as ours, employing machine learning and deep learning techniques.
Authors in [127] conducted a comparative analysis of machine learning-based website phishing detection. They assessed five machine-learning techniques: Decision Tree (DT), Random Forest (RF), K Nearest Neighbors (KNN), Gaussian Naïve Bayes (GNB), and XGBoost. The dataset used for their study was "Web page phishing detection" (2021) from the Mendeley Data repository, containing 11,430 URLs with 87 extracted features. Among these features, 56 were URL-based, 24 were obtained from the content of related pages, and 7 were external service-based. For preprocessing, they employed maximum relevance-minimum redundancy to select the most important 30 features out of the 87. In evaluating performance, they utilized metrics such as Accuracy, Sensitivity, Specificity, F1-score, and area under the curve (AUC). The results indicated that the Random Forest algorithm outperformed the other proposed algorithms. Authors in [88]utilized the "Web page phishing detection" dataset (2021) sourced from the Mendeley Data repository, consisting of 11,430 samples and 87 extracted features. This dataset serves as a standard benchmark for machine learning-driven phishing detection systems. Employing an ensemble learning technique called Gradient Boosting, they trained their classification model. The researchers refined the model through hyperparameter tuning. Their evaluation encompassed various metrics including accuracy, precision, recall, F1-score, and the area under the curve. Results revealed that the optimized ensemble model achieved an accuracy rate of 95.36% in distinguishing between legitimate and phishing URLs. In their study described in [112], the authors introduced a Cloud Machine Learning approach aimed at evaluating and gauging the time needed for detecting website phishing. They employed three pre-existing algorithms within SageMaker: XGBoost, Linear Learner, and KNN. The dataset utilized in this research, "Web page phishing detection" (2021), was retrieved from the Mendeley Data repository and comprised 11,430 samples featuring 87 extracted attributes. During preprocessing, the study leveraged Pearson Correlation to select features deemed significant. The correlation analysis identified 9 features exhibiting correlations exceeding 70%, indicating a substantial interrelationship among these attributes. Performance assessment entailed evaluating metrics such as Accuracy, Precision, F-1 score, Recall, and Mislabeling. Findings revealed that Extreme Gradient Boosting surpassed the other two algorithms, achieving an accuracy of 96.4% with an online prediction time of 0.0005 minutes. In their work outlined in [47], the authors introduced a method aimed at identifying the optimal architecture for an artificial neural network (ANN) by employing a bio-inspired metaheuristic algorithm known as the Water Cycle Algorithm (WCA). They applied this approach to the Phishing Websites Dataset 2020 obtained from the Mendeley Data repository, which encompasses 88,647 samples and 111 features. By varying the number of hidden layers and adjusting the neuron count accordingly, they exclusively assessed accuracy as the performance metric. Results revealed that the highest accuracy attained with this dataset was 94.03%, achieved with a single hidden layer comprising 78 neurons. In their research presented in [1], the authors utilized the Binary Slap Swarm Optimiza-

tion Algorithm (BSSA) in conjunction with various transfer functions (TFs), including S-shaped, U-shaped, V-shaped, X-shaped, and Z-shaped TFs. Their experimentation was conducted on the Phishing Websites Dataset 2020 sourced from the Mendeley Data repository, which comprises 88,647 samples and 111 features. Through their analysis, the authors identified an optimal subset of 49 features from the initial 111 features. The study's performance evaluation focused on accuracy and fitness value metrics. Results indicate that employing the BSSA with the X-shaped TF, followed by KNN, yielded superior performance compared to other TFs, achieving an accuracy rate of 95.07%. In their study outlined in [24], the authors introduced a feature selection approach known as Differential Evolution with a Threshold Mechanism (DEFSTH). They applied this method to the Phishing Websites Dataset 2020 sourced from the Mendeley Data repository, which consists of 88,647 samples and 111 features resulting in 40 features. The classifiers employed in their analysis included Logistic Regression (LG), Naïve Bayes (NB), KNN, DT, and Multilayer Perceptron (MLP). Performance evaluation was based on metrics such as accuracy, F1 score, and area under the curve (AUC). The proposed method, when combined with a Naive Bayes classifier, yielded an accuracy rate of 96.82%. In their work outlined in [49], the authors presented a URL phishing detection approach employing BERT feature extraction coupled with a deep learning methodology. BERT was harnessed to extract textual content from URLs within the Phishing Site Predict dataset. Subsequently, a deep convolutional neural network method was employed to identify phishing URLs. The dataset utilized for this study, sourced from Kaggle, comprised 549,346 URLs categorized as either good or bad. Performance metrics such as Accuracy, F1-Score, and Precision were utilized. The proposed method achieved an accuracy rate of 96.66%. In their study [78], the authors introduced a machine learning-driven approach aimed at identifying malicious websites. They utilized the "Phishing site Predict dataset" sourced from Kaggle, comprising 549,346 URLs categorized as either benign or malicious. Preprocessing involved the application of three text feature extraction techniques: count vectorizer, hashing vectorizer with Inverse Document Frequency (IDF) vectorizer, and subsequently, the development of a phishing website detection model employing four machine learning classifiers: Logistic Regression, K-NN, Decision Tree, and Random Forest. The model employing the hash vectorizer and Random Forest classifier achieved an impressive accuracy of 97.5%. Additionally, they deployed their model as a web application. The authors in [63] introduced a model designed to identify malicious URLs utilizing the CNN deep learning technique. They employed the Malicious URLs dataset, publicly accessible on Kaggle, comprising 651,191 labeled URLs. During preprocessing, they extracted domains, subdomains, and domain suffixes from the URLs, encoding them into integer values. Additionally, they standardized the length of input URLs through padding and tokenization, preparing them as inputs for the CNN-based deep learning model. Performance evaluation relied on precision, recall, and F1-score metrics. In their study [69], the authors implemented machine learning algorithms, specifically Random Forest, Light Gradient Boosting Machine (LightGBM), and XGBoost, to detect fraudulent websites. They utilized the Malicious URLs dataset, available on Kaggle, which comprised 651,191 labeled URLs classified into four categories: benign, defacement, phishing, and malware. During preprocessing, they extracted various features from website content and metadata to train and evaluate the algorithms. Performance evaluation was based on

metrics such as Accuracy, Precision, Recall, and F1-score. The findings revealed that Random Forest exhibited superior accuracy compared to both LightGBM and XGBoost. Adebowale et al. [2] implemented an Intelligent Phishing Detection System (IPDS) aimed at identifying phishing URLs. The IPDS incorporates a hybrid classification model employing LSTM and CNN. They utilized 1 million URLs and over 10,000 images for model training. In the preprocessing phase, images were cropped from the websites based on bounding boxes, and erroneous images were removed for the CNN architecture. For the LSTM, various website features were extracted. Performance metrics included Accuracy, Recall, Precision, and F-measure. Their system achieved a classification accuracy of 93.28%. Aljofey et al. [7] proposed a deep learning-based solution model for phishing detection focusing on website URLs, utilizing a character-level convolutional neural network. The dataset comprised 318,642 URLs, with 157,626 benign and 161,016 phishing URLs. In preprocessing, they employed a sequential pattern to capture URL information, utilizing these sequential pattern features for rapid classification of actual URLs. Performance metrics included Accuracy, Precision, and F-measure, with the model achieving an accuracy of 95.02% on their dataset. Le et al. [82] introduced a method, termed URLNet, aimed at identifying malicious URLs. Their approach utilized convolutional neural networks (CNN) to analyze both the characters and words within the URL string to derive its embedding. In the preprocessing phase, the raw URL input underwent processing via two branches: a character-level branch and a word-level branch. The Character CNNs operated directly at the character level, thus obviating the need for feature extraction, whereas, for the Word CNN, lexical features were extracted in the form of a Bag of Words. Additionally, manual expert features were incorporated and integrated into the overall model. The effectiveness of their method was evaluated using a large-scale dataset comprising 1 to 5 million labeled URLs sourced from VirusTotal. Performance evaluation was conducted using key metrics such as True Positive Rate (TPR), False Positive Rate (FPR), and AUC. Alshingiti et al. [9] introduced a deep learning-driven system for detecting phishing attempts, incorporating three distinct deep learning models: LSTM, CNN, and a hybrid LSTM-CNN model. They gathered a dataset comprising 20,000 samples from URL 2016|Datasets|Research|Canadian Institute for Cybersecurity|UNB. Unb.ca. 2022. During the preprocessing phase, they employed the SelectKBest method to identify the 30 most effective features. Performance assessment was based on metrics including accuracy, precision, recall, and F1-score. The evaluation demonstrated that the CNN model exhibited superior accuracy compared to the other models. Opara et al. [96] introduced a deep learning-based approach called HTMLPhish for detecting phishing webpages. They utilized two datasets: D1, comprising 23,000 legitimate URLs and 2,300 phishing URLs, and D2, consisting of 24,000 legitimate URLs and 2,400 phishing URLs. Legitimate URLs were sourced from Alexa.com while phishing URLs were gathered from Phishtank.com. In the preprocessing phase, they employed two scenarios: word and character embedding, followed by concatenation of the two embeddings to serve as input to a CNN for learning semantic dependencies in the textual contents of the HTML for classification. Performance metrics included Accuracy, Precision, True Positive Rates, F-1 Score, AUC, training, and testing time. Their results demonstrated over 93% Accuracy and True Positive Rate.

3.7.1 ***Summary***

Table 3.2 provides a recapitulation of the related studies

Table 3.2: Comparison of the related works.

Ref/Year	Dataset	Preprocessing techniques	Applied Models	Performance metrics used	The use of parameters tuning	Adv	Disadv
[127]/2022	« Web page phishing detection»(2021) from the Mendeley Data. 11,430 URLs with 87 features	maximum relevance-minimum redundancy for features selection	Decision Tree, Random Forest, KNN, Gaussian Naïve Bayes, and XGBoost	Accuracy, Sensitivity, Specificity, F1-score, and AUC		the utilization of the three types of features, which enhances phishing detection	Inability to overcome new phishing attack tactics.

Continued on next page

Table 3.2 – *Continued from previous page*

Ref/Year	**Dataset**	**Preprocessing techniques**	**Applied Models**	**Performance metrics used**	**The use of parameters tuning**	**Adv**	**Disadv**
[88]/2023	« Web page phishing detection»(2021) from the Mendeley Data. 11,430 URLs with 87 features	/	Gradient Boosting	accuracy, precision, recall, F1-score, and the AUC	✓	the utilization of the three types of features and the optimization of hyperparameters, which enhances phishing detection.	Inability to overcome new phishing attack tactics.
[112]/2023	« Web page phishing detection»(2021) from the Mendeley Data. 11,430 URLs with 87 features	Pearson Correlation for features selection	XGBoost, Linear Learner, and KNN	Accuracy, Precision, F-1 score, Recall, and Mislabeling.		The utilization of the three types of features and the minimization of prediction time enhances phishing detection.	Inability to overcome new phishing attack tactics.

Continued on next page

Table 3.2 – *Continued from previous page*

Ref/Year	**Dataset**	**Preprocessing techniques**	**Applied Models**	**Performance metrics used**	**The use of parameters tuning**	**Adv**	**Disadv**
[47]/2022	« Phishing Websites Dataset 2020 » from Mendeley data that contains 88647 samples and 111 features.	/	Artificial Neural Network (ANN)	Accuracy	Optimizing only the architecture of the ANN	The utilization of deep learning techniques and the optimization of the ANN topology	The absence of feature preprocessing, the hyperparameter tuning and the exclusive reliance on accuracy as the sole metric to evaluate the model. The non-use of the balancing technique.
[1]/2021	« Phishing Websites Dataset » 2020 from Mendeley data that contains 88647 samples and 111 features	BSSA for features selection	BSSA-X+KNN	accuracy, and fitness value		the utilization of an optimization algorithm for feature selection	Inability to overcome new phishing attack tactics and reliance solely on two metrics, accuracy, and fitness value, to evaluate the model. The non-use of balancing technique

Continued on next page

Table 3.2 – *Continued from previous page*

Ref/Year	**Dataset**	**Preprocessing techniques**	**Applied Models**	**Performance metrics used**	**The use of parameters tuning**	**Adv**	**Disadv**
[24]/2019	« Phishing Websites Dataset » 2020 from Mendeley data that contains 88647 samples and 111 features	a feature selection approach known as Differential Evolution with a Threshold Mechanism (DEFSTH)	Logistic Regression, Naive Bayes, k-Nearest Neighbors, Decision Tree, and Multilayer Perceptron	accuracy (ACC), F1 score, and AUC		the utilization of the feature selection method with a threshold	Inability to overcome new phishing attack tactics.
[49]/2022	the "Phishing site Predict dataset" from Kaggle that contains 549,346 URLs labeled as good or bad.	BERT as a features extractor	CNN	Accuracy, F1-Score, and Precision		the combination of deep learning and NLP to detect phishing URLs	The training time is too long.

Continued on next page

Table 3.2 – *Continued from previous page*

Ref/Year	**Dataset**	**Preprocessing techniques**	**Applied Models**	**Performance metrics used**	**The use of parameters tuning**	**Adv**	**Disadv**
[78]/2021	the "Phishing site Predict dataset" from Kaggle that contains 549,346 URLs labeled as good or bad.	three text feature extraction techniques: count vectorizer, hashing vectorizer with IDF vectorizer	Logistic Regression, K-NN, Decision Tree, and Random Forest	Accuracy		the development of the web application, enhancing user-friendliness and delivering a final product to web users	drawbacks include the limitations associated with the hashing vectorizer. Specifically, the hashing vectorizer fails to retain the subsequent vocabulary in memory, resulting in prolonged processing times, the creation of large and sparse matrices, and the loss of semantic word meanings.

Continued on next page

Table 3.2 – *Continued from previous page*

Ref/Year	**Dataset**	**Preprocessing techniques**	**Applied Models**	**Performance metrics used**	**The use of parameters tuning**	**Adv**	**Disadv**
[63]/2023	Malicious URLs dataset from Kaggle consisting of 651,191 URLs, out of which 428103 benign or safe URLs, 96457 defacement URLs, 94111 phishing URLs, and 32520 malware URLs.	Features extraction, padding, and tokenization of the input URL.	CNN	precision, recall, and F1-score metrics		the implementation of Deep Learning techniques on a large-scale dataset	working with an unbalanced dataset, not utilizing accuracy as a performance metric, and not employing parameter tuning for optimization. the number of features is not enough for detection and the inability to overcome new phishing attack tactics

Continued on next page

Table 3.2 – *Continued from previous page*

Ref/Year	**Dataset**	**Preprocessing techniques**	**Applied Models**	**Performance metrics used**	**The use of parameters tuning**	**Adv**	**Disadv**
[69]/2023	Malicious URLs dataset from Kaggle consisting of 651,191 URLs, out of which 428103 benign or safe URLs, 96457 defacement URLs, 94111 phishing URLs, and 32520 malware URLs.	Features extraction from website content and metadata.	Random Forest, LightGBM, and XGBoost	Accuracy, Precision, Recall, and F1-score		providing a final product for end users with real-time prediction capabilities and results, as well as the selection of a large-scale dataset	dealing with an unbalanced dataset and the inability to overcome new phishing attack tactics.

Continued on next page

Table 3.2 – *Continued from previous page*

Ref/Year	**Dataset**	**Preprocessing techniques**	**Applied Models**	**Performance metrics used**	**The use of parameters tuning**	**Adv**	**Disadv**
[2]/2020	1 million URLs/ 10.000 images from PhishTank and WHOIS	Features extraction from URL	CNN, LSTM	Accuracy, Precision, Recall, and F1-score		the use of DL techniques and the online plugin model to provide a final product for end users, the choice of the varied and the large scale dataset	the low accuracy value, the non-use of optimization by the parameters tuning and vulnerability to new phishing attack tactics.

Continued on next page

Table 3.2 – *Continued from previous page*

Ref/Year	**Dataset**	**Preprocessing techniques**	**Applied Models**	**Performance metrics used**	**The use of parameters tuning**	**Adv**	**Disadv**
[7]/2020	318,642 URLs 157,626 benign and 161,016 phishing from Alexa, Openphish, spamhaus.org, tech-helplist.com, isc.sans.edu, and PhishTank	employing a sequential pattern to capture URL information	CNN	Accuracy, Precision, Recall, and F1-score		the use of deep learning techniques, and the non-need for retrieving target website content or relying on third-party services	the lack of optimization through parameter tuning and dealing with an imbalanced dataset.

Continued on next page

Table 3.2 – *Continued from previous page*

Ref/Year	**Dataset**	**Preprocessing techniques**	**Applied Models**	**Performance metrics used**	**The use of parameters tuning**	**Adv**	**Disadv**
[82]/2018	1 /5 million labeled URLs from VirusTotal.	character-level branch and a word-level branch. lexical features extracted in the form of a Bag of Words. Additionally, manual expert features were extracted.	CNN	TPR, FPR, AUC		the use of both the URL characters and words embedding with the deep learning techniques on a large scale dataset	the lack of optimization through parameter tuning and dealing with an imbalanced dataset and evaluating the model with only AUC, TPR, and FPR metrics.
[9]/2023	20,000 samples from URL 2016|Datasets|Canadian Institute for Cybersecurity 2022.	SelectKBest for features selection	CNN, LSTM and LSTM-CNN	Accuracy, Precision, Recall, and F1-score	✓	the use of three DL models	working with an unbalanced dataset and the training process is a bit long.

Continued on next page

Table 3.2 – *Continued from previous page*

Ref/Year	**Dataset**	**Preprocessing techniques**	**Applied Models**	**Performance metrics used**	**The use of parameters tuning**	**Adv**	**Disadv**
[96]/2020	D1 with 23,000 legitimate URLs and 2,300 phishing URLs. D2 with 24,000 legitimate URLs and 2,400 phishing URLs. sourced from Alexa.com and Phishtank.com.	Charatcter and word embedding	CNN	Accuracy, Precision, True Positive Rates, F-1 Score, AUC		the utilization of DL model techniques with HTML content and context features, training and testing times are reduced	working with an unbalanced dataset, rendering the results not reliable, and relying solely on one type of feature, which may lead to this approach being bypassed by phishers who exploit other features (inability to overcome new phishing attack tactics).

After a comprehensive analysis of the related works, it is noticed that ML and DL are commonly utilized in detecting phishing websites. However, deep learning is increasingly becoming prominent in this field, offering a solution to the limitations of traditional anti-phishing methods. According to Table 3.2, the outcomes of these investigations demonstrate that deep learning approaches achieve accuracy levels of 90% or higher in phishing detection.
LSTM and CNN have been extensively employed in prior research, largely due to their manifold advantages that demonstrate impressive outcomes. LSTM models address issues like vanishing or exploding gradients present in traditional recurrent neural networks, making them adept at handling time-series sequence data and proficient in capturing extended relationships within sequential data. Conversely, CNN models excel at efficient and rapid feature extraction from intricate and unprocessed data. Their architectures yield more promising and resilient results by reducing network complexity, accelerating the learning process, and adeptly discerning local patterns in data. These attributes render LSTM and CNN particularly well-suited for detecting phishing webpages, given the multidimensional nature of such websites, which often comprise text, images, or both. Integration of these models with heuristic techniques allows for the utilization of URL features as inputs in deep learning classifiers. Notably, studies have consistently demonstrated the efficacy of URL features such as URL length, character composition, frequency of keywords, and occurrence of suspicious symbols, particularly when assessed on datasets collected from platforms like VirusTotal, PhishTank, OpenPhish, and other open phishing databases. We also noted that all CNN-based studies utilized 1D CNN architecture. The utilization of 2D CNN is less frequent in the phishing domain.
Phishing detection research presents a formidable challenge, as cybercriminals continuously devise sophisticated methods to evade existing detection systems. Moreover, studies employing deep learning (DL) for phishing website detection encounter numerous challenges, and their performance hinges on several critical factors:

- Preprocessing Techniques: The selection of preprocessing techniques, including feature extraction and selection, significantly impacts the subsequent stages of the DL model. Extracting relevant features and employing appropriate statistical techniques for feature reduction are pivotal for model efficacy.

- Choice of DL Model: The diverse array of DL techniques available for phishing attack detection underscores the importance of selecting the most suitable algorithm. The ultimate performance of the model is heavily influenced by this choice, emphasizing the need for careful selection.

- Dataset Quality: The effectiveness of DL models relies heavily on the diversity, quality, and availability of datasets. Previous works have encountered issues with non-representative, irrelevant features, and imbalanced datasets, leading to results that lack relevance.

- Evaluation Metrics: Comprehensive evaluation metrics are essential for accurately assessing

the performance of a phishing detection system. While a single metric may not adequately represent the performance of a DL algorithm, some studies overlook the computation of all relevant performance measures. Utilizing appropriate evaluation metrics is crucial for meaningful assessment.

- Parameter Optimization: The performance of DL models is intricately tied to the optimization of parameters such as the number of epochs, type of activation function, number of neurons in each layer, learning rate, and type of optimizer. Fine-tuning these parameters through a series of experiments is time-consuming and resource-intensive. Unfortunately, many previous studies have overlooked this critical step in their DL-based phishing detectors.

- Training Time: DL models demand a substantial amount of data and time for training. The prolonged training duration poses a limitation in several prior studies.

Considering these factors is imperative in the realm of DL-based phishing detection. Therefore, there is a pressing need for a system capable of efficiently and effectively identifying phishing URLs.

3.8 Conclusion

In conclusion, this chapter has provided a comprehensive literature survey on the persistent threat of phishing attacks to web users' information security. We began by defining phishing attacks and elucidating the steps involved in their execution. Subsequently, we explored the motives and statistics behind phishing, followed by an examination of the various types of phishing attacks prevalent on the web. Additionally, we outlined a classification of phishing detection mechanisms found in existing literature, aimed at safeguarding users against these cyber threats. Furthermore, we reviewed significant works in the field. It is evident that detecting phishing remains a formidable challenge due to the dynamic nature of attackers' strategies; however, this challenge has not deterred researchers from pursuing innovative solutions. Despite the evolving tactics employed by phishers, ongoing research efforts leverage emerging technologies to develop effective systems capable of identifying phishing URLs efficiently. In the forthcoming chapter, we will present our approach to detecting phishing websites, aiming to address the limitations identified in existing literature. We have opted to focus on website-based attacks due to their widespread prevalence and their role as a common vector for other types of phishing attacks, such as email and phone scams. By targeting phishing websites, we aim to provide a comprehensive solution that addresses various manifestations of this cyber threat.

Chapter. 4

Contributions to Cybersecurity: Deep Learning for Phishing Websites Detection

4.1 Introduction

In the field of cybersecurity, Deep Learning has assumed growing significance in recent years, empowering various applications and strategies to effectively combat numerous security threats. Particularly in the realm of website phishing detection, several approaches have achieved successful results despite the challenges discussed in the previous chapter. Therefore, the development of robust DL-based systems for phishing website detection is an important area of research that has the potential to significantly reinforce cybersecurity and protect the sensitive data of web users.
Our study is based on this technology, for which we will present, in this chapter, the fundamental elements and terminology necessary for its understanding. Subsequently, we will establish the deep learning algorithms used, the databases utilized, and the performance measurement techniques adopted in our research. Furthermore, we aim to highlight our contributions to cybersecurity within the context of applying DL to detect phishing websites.
This chapter is organized as follows: Section 4.2 presents the background and general architecture for applying deep learning to detect phishing websites. Following that, Section 4.3 discusses our first contribution, which involves detecting phishing websites using 1D and 2D CNN models. Then, Section 4.4 outlines our second approach for detecting phishing websites, utilizing the permutation importance method alongside various distinct classifiers. Finally, Section 4.5 concludes this chapter.

4.2 Background and general architecture

4.2.1 Steps of applying DL to detect phishing websites

The machine learning life cycle encompasses various stages, starting from data collection to model predictions, as follows [16], [71]: The detection of phishing websites based on deep learning consists of several processes as shown in Figure 4.1, which details the overall framework of our phishing website detection system starting from data collection to model predictions, as follows:

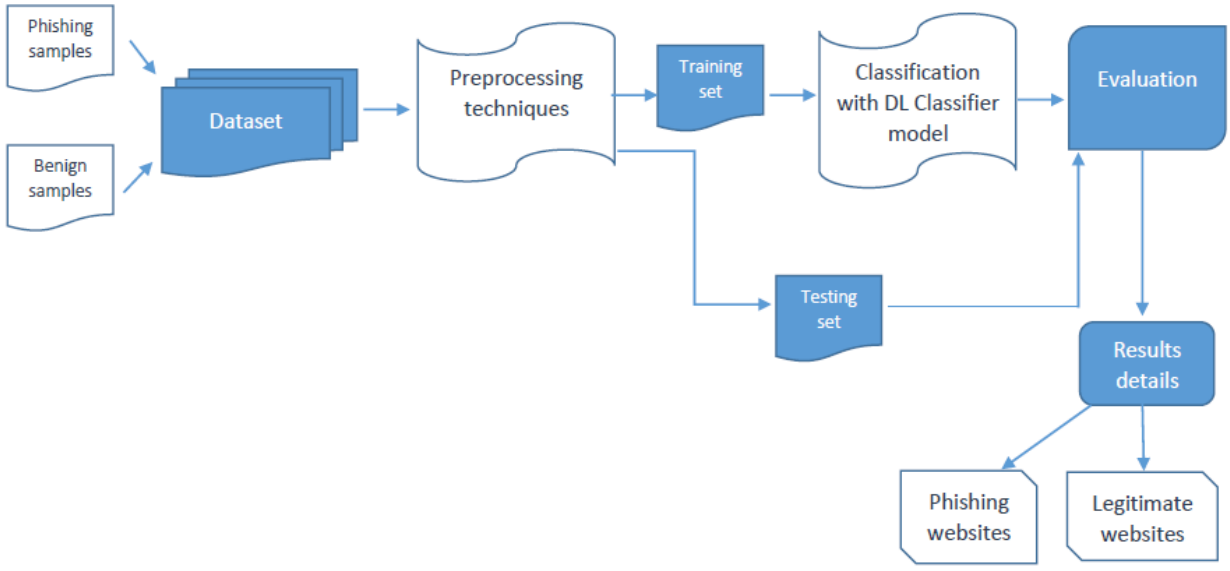


Figure 4.1: General diagram of the proposed DL-based system to detect phishing websites.

4.2.1.1 Data collection & datasets used in our study

Gathering the data is the initial and crucial phase in deep learning. This stage holds significant importance as the effectiveness and strength of the predictive model directly hinge upon the quality and quantity of the data acquired. The gathered data will serve as the foundation for creating a dataset, which is essential for the training process. The selection and quantity of data required are contingent upon the specific problem at hand [16], [71].
In our thesis, we decided to use publicly available databases as they will be useful for comparison purposes. From the literature review study, we used four datasets. Each of them consists of two parts, one composed of phishing website data and the other containing legitimate website data. The four datasets used are:

1. **«Web page phishing detection dataset» from Mendeley Data 2021 [60]**
   This is a publicly available dataset from Mendeley data repository [60]. It was compiled from diverse sources, including Alexa and PhishTank, and is presented in the form of a Comma-Separated Values (CSV) file. This file encompasses 87 features along with their corresponding URLs, amounting to a total of 11,430 samples. Notably, the dataset is well-balanced, consisting of an equal distribution of 50% phishing URLs and 50% legitimate URLs. The features are categorized into three distinct classes: 56 features extracted from the structure and syntax of URLs, 24 features extracted from the content of their respective pages, and 7 features obtained through queries to external services [59]. Further details about these three feature types are elaborated upon in the subsequent items.

   - **URL-based features**
     A URL, or Uniform Resource Locator, is essentially the digital address for an online resource, such as a website or document. Figure 4.2 depicts a detailed structure of URL [114]. URL-based features are obtained through the examination of the textual content within URLs. These features can be further classified into statistical and structural categories. Statistical-based features are determined by the quantity or distribution of elements such as specific words or characters in the URL text. Illustrative features in this category encompass the count of digits in the URL, the number of subdomains, and the word length. On the other hand, structural-based features focus on the position, presence, and characteristics of elements within the URL, including path, port, protocol, domain, subdomains, and top-level domain. Examples of such features include the presence of a prefix or suffix in the URL domain name and the inclusion of the Hypertext Transfer Protocol Secure (HTTPS) protocol in the URL. Within the used dataset, a total of 56 features are derived from the structure and syntax of URLs, and these individual characteristics are detailed in Table 4.1 [59].

Figure 4.2: Parts of a URL structure [114].

   - **Content-based features**
     Content-based features consist of features derived from the content of the web pages and their HTML contents. In the dataset employed for this study, a set of 24 features

is extracted from the content of the respective pages. A detailed description of each of these features can be found in Table 4.2 [59].

- **Third-party services-based features**
  Third-party features involve the utilization of services like WHOIS, Alexa, and search engines such as openpagerank and Google for their extraction. Within the experimental dataset, a total of 7 features are acquired through queries to external services [59]. The details of these features are provided in Table 4.3

Table 4.1: Description of the URL-based features [107], [59].

Name of Feature	Type	Definition
length_url	Int	Calculates the character count of different parts of the URL (total length and length of the hostname). Phishers frequently employ lengthy URLs to conceal authentic domain names.
length_hostname		
Special Characters (nbr_dots, nbr_hyphens, nbr_at, nbr_qm, nbr_and, nbr_or, nbr_eq, nbr_underscore, nbr_tilde, nbr_percent, nbr_slash, nbr_star, nbr_colon, nbr_comma, nbr_semicolumn, nbr_dollar, nbr_space)	Int	Count the number of special characters in the URL such as (dots, hyphens, percent...) that are an indicator of a phishing URL.
ip	Binary (0/1)	It verifies whether the URL contains an IP address. Including an IP address instead of a hostname is typically regarded as a phishing tactic.

Continued on next page

nbr_www, nbr_com, nbr_dslash, http_in_path	Int	These features count the occurrences of common terms in URLs, such as 'www,' ,'.com,' 'http', and '//.' An occurrence of these terms more than once in the URL serves as a phishing indicator.
ratio-of_digits-in_url	Float	Count the number of digits in the domain name and hostname. a substantial quantity of numerical digits within URLs is regarded as an indication of phishing.
ratio-of_digits-in_host		
https_token	Binary	It examines the presence of the HTTPS protocol in the URL. The utilization of HTTPS serves as an indicator of legitimacy.
prefix_suffix	Binary	Verifies if the domain name contains prefixes and suffixes, indicating it might be a phishing link.
Port	Binary	Examines if the URL includes a port number, which is considered indicative of phishing activity.
Punycode	Binary	URLs containing puny-codes are deemed as phishing. Punycode is employed in domain names to substitute certain ASCII characters with Unicode characters, making URLs appear authentic even when referring to distinct websites.
nb_subdomains	Int	It counts how many subdomains are included in the URL where more numbers of subdomains is a phishing indicator.
abnormal_subdomain	Binary	malicious URLs might employ the pattern 'w[w]?[0-9]*' instead of 'www' to trick users. Consequently, URLs with subdomains that match this pattern are classified as phishing.

Continued on next page

tld_in_subdomain	Binary	Check the position of the TLD, If toplevel domains (TLDs) are present in the path or subdomain section, the URL is regarded as phishing.
tld_in_path		
path_extension	Binary	It checks the presence of harmful path extensions such as: 'txt', éxe,' 'js.' Legitimate pages can be compromised by the addition of malicious scripts. Certain file extensions included in URL paths may initiate such attacks.
suspecious_tld	Binary	TLDs (Toplevel domains) are checked for suspiciousness using a list of suspicious TLDs collected from: Spamhaus.org and Blue coast system Inc (https://www.broadcom.com/).
shortening_service	Binary	Using a URL shortening service is regarded as a phishing indicator since this service can be employed in phishing attempts to conceal the actual hostname.
random_domain	Binary	It examines domain names for the presence of random characters, which serve as a phishing indicator in the URL.
nb_redirection	Int	URLs can undergo redirection either to pages within the same domain (internal redirection) or to pages hosted on different domains (external redirections). These features Count the number of redirections and external redirections, serving as phishing indicators, as redirections can be used for malicious purposes.
nb_external_redirection		
statistical_report	Binary	The IP addresses of URL domains are verified against a compilation of top phishing domains obtained from prior research.

Continued on next page

length-of_words_raw chars_repeat shortest-of_words_raw shortest-of_word_host shortest-of_word_path longest-of_words_raw longest-of_word_host longest-of_word_path avrg_words_raw	Int	These features take into account factors such as the number of words, character repetitions, the shortest words in URLs, hostnames, and paths, the longest words in URLs, hostnames, and paths, as well as the average length of words in URLs, hostnames, and paths. Since phishing detection involves the utilization of natural language processing and word-related features.
avrg-of_word_host avrg-of_word_path	Float	

Continued on next page

if-domain-is_in_brand if-brand-is_in_subdomain if-brand-is_in_path	Binary	Check the occurrence of brand names in the domain, subdomain, and paths, as phishing URLs often incorporate brand domain names in various URL segments. The presence of brand names in the domain part is indicative of legitimacy, whereas their presence in subdomains or paths is regarded as a phishing indicator.
phish_hints	Int	It counts the number of sensitive words(such as ‘admin’, ‘login’, ‘myaccount’, ‘alibaba’..etc) in URLs that is considered as phishing indicators.

Table 4.2: Description of the Content-based features [107], [59].

Name of Feature	Type	Definition
nb_hyperlinks	Int	Tabulates the number of links within the webpage URL content, as phishing sites typically have fewer links.
ratio_intHyperlinks/ ratio_extHyperlinks	Float	Count the ratios of internal and external hyperlinks found on web pages. Phishing pages tend to feature numerous external hyperlinks directing users to target websites.
popup_window	Binary	Checks whether pop-up windows containing text fields are present, as they are regarded as an indicator of phishing activity.
ratio_nullHyperlinks	Float	Calculates the ratio of empty hyperlinks within tags, which serves as a phishing indicator as they are often utilized to imitate target pages.
nb_extCSS	Int	Counts the quantity of external Cascading Style Sheets (CSS) files, a potential phishing indicator, as phishing websites often employ only one external CSS file containing links to CSS files from target websites.
ratio_int Redirection ratio_ext Redirection	Float	Count ratios of internal and external redirections because these links may redirect to other legitimate or fake pages.
login_form	Binary	Login forms featuring external action links or empty actions are classified as phishing attempts.
links_in_tags	Float	Verify the ratio of internal links within <Link> tags, as it is utilized in phishing detection.

Continued on next page

external_favicon	Binary	It verifies whether the favicon is loaded from an external domain, as websites employing external favicons are flagged as potential phishing sites.
submit_email	Binary	Check for Form actions containing 'mail to:' or 'mail()' that are a phishing indicator.
ratio_intErrors ratio_extErrors	Float	Count ratios of internal and external hyperlinks connection errors, since are usually used in malicious websites.
ratio_intMedia ratio_extMedia	Float	Count the ratios of internal and external media file links to be used for identifying legitimate from phishing websites. The phishing websites tend to rely on a greater amount of external media, often stored within the domain of the targeted website, as a means of conserving storage space.
safe_anchor	Int	Counts the number of unsafe anchors on the webpage like tags with such links ‘’, ‘javascript’, ‘mailto’.
sfh	Binary	Check webpage forms containing an empty string or 'about:blank', which are identified as components of phishing web pages.
right_clic onmouseover	Binary	Check for the presence of the "onmouseover" feature and the disabling of the right-click function, as these are considered phishing indicators. Phishers may use these techniques to prevent the viewing of the source code of web pages.

Continued on next page

domain_with_copyright	Binary	Verify the existence of the domain within the copyright logo, as this serves as an indicator of legitimacy.
domain_in_title	Binary	Check for the presence of the domain of URL as part of the web page title as this serves as an indicator of legitimacy.
empty_title	Binary	Check for the presence of the webpage title in the <title> tag, since the absence of a web page title is considered a phishing indicator.
iframe	Binary	Check for invisible <iframe> tags that is considered as a phishing indicator.

Table 4.3: Description of the features based on the third-party services [107], [59].

Feature name	Type	Description
whois_registered_domain	Binary	Checks for domain registration in whois database. URLs featuring domains not registered in WHOIS are deemed malicious.
domain_age	Int	Determines the domain age of the website since the phishing web pages have fewer domain ages.
web_traffic	Int	Identifying the web traffic of URLs by Alexa website since contrary to legitimate websites, phishing websites usually have fewer visitors.
dns_record	Binary	It verifies whether the URL is registered in the DNS, as the absence of a DNS record serves as a phishing indicator.
google_index	Binary	Check if the web page is indexed by google since phishing websites are generally not indexed by google because they are short-lived.
page_rank	Int	Getting the page rank value by the Openpagerank. In contrary to a legitimate webpage, Phishing websites have low page ranks.
domain_registration_length	Int	Calculates the duration for which the domain renewal fee has been paid, as it serves as a phishing indicator. Phishing websites typically have short lifespans and are unable to be renewed for extended periods.

2. **«Phishing Websites Dataset» from Mendeley Data [133]**
   The Mendeley phishing website datasets, as referenced in [133], are divided into two variants labeled D1 and D2. D1 contains 111 features and 58,645 instances, while D2 also consists of 111 features but with 88,647 instances. Each dataset is characterized by two classes: phishing and legitimate. D1 serves as the smaller version, comprising 27,998 instances of legitimate websites marked as 0 and 30,647 instances of phishing websites marked as 1. On the other hand, D2 represents the full phishing dataset, with 58,000 instances of legitimate websites labeled 0 and 30,647 instances of phishing websites labeled 1. Legitimate URLs were sourced from the Alexa ranking website while phishing URLs were obtained from PhishTank. This dataset includes URL-related attributes, URL resolution metrics, and features based on external services.
   The dataset's URL characteristics are derived by segmenting the URL into five components: Domain, Directory, File, Parameters, and the complete URL. Within each segment, the count of specific symbols is tallied to form features. These 111 features across the datasets are further organized into 8 distinct groups, provided in Table 4.4, describing the features used. The feature names of this dataset are provided in Table A.1.

Table 4.4: Features description of the Phishing Websites Dataset 2 [133].

Group	Feature	Definition	Type
1	F1 → F17	The total count of each symbol such as: .-_/ ?=@&! ∼ ,+*# ”$% present in the entire URL.	Numeric
2	F20 → F36	The total count of each symbol such as: .-_/ ?=@&! ∼ ,+*# ”$% present in the URL domain.	
3	F41 → F57	The total count of each symbol such as: .-_/ ?=@&! ∼ ,+*# ”$% present in the domain in directory	
4	F59 → F75	The total count of each symbol such as: .-_/ ?=@&! ∼ ,+*# ”$% present in the domain in file	
5	F77 → F93	The total count of each symbol such as: .-_/ ?=@&! ∼ ,+*# ”$% present in the domain in parameters	
6	F37, F96, F98, F100 → F106, F108	vowels count, count of parameters, time_response, asn_ip, time_domain_activation, time_domain_expiration, count of resolved Ips, count of resolved NS, number of MX servers, Time-To-Live, count of redirects	Numeric
7	F18, F19, F38, F58, F76, F94	length of Top-level domain character, count of characters in the entire URL, count of domain characters, count of directory characters, count of file characters, count of parameters characters	Numeric
8	F97, F39, F40, F95, F99, F107, F109 → F111	Is there an email included, is the URL's domain formatted as an IP address, does the domain include "server" or "client", is the top-level domain included in the parameters, does the domain have Sender Policy Framework (SPF), does the URL have a valid Transport Layer Security (TLS)/Secure Sockets Layer (SSL) certificate, is the URL indexed on Google, is the domain indexed on Google, is the URL shortened	Boolean (Binary)

3. **«Phishing Site URLs dataset» from Kaggle [125]**
   This dataset was gathered from the public and renowned repository Kaggle.com [125]. Named the "Phishing Site URLs Dataset", it comprised 549,346 entries across two columns. One column contained the website URLs, while the other column contained the website labels, categorized as either good (benign site) or bad (phishing site). Within this dataset, there were 392,924 samples classified as belonging to the good class and 156,422 samples classified as belonging to the bad class. Table 4.5 presents a small sample of the chosen dataset:

Table 4.5: Example Subset of «Phishing Site URLs dataset» [125].

URL	Label	ID
listafterlist.com/tabid/57/listid/9207/News++W...1	good	378189
bit.ly/2bYh97s	bad	533731
themuseinmusic.com/2011/05/04/tmimreviewsla...	good	246949
linsy.eu/tmp/dpbx	bad	120685
thamtutuuytin.com/kn4mdmsfk	bad	544999

4. **« Malicious URLs dataset» from Kaggle [118]**
   This dataset, obtained from Kaggle [118] and titled Malicious URLs, comprises 651,191 URLs classified into four categories: Benign, Malware, Phishing, and Defacement URLs. The majority of URLs (428,103) are categorized as Benign, with the remaining URLs classified as 94,111 Phishing URLs, 96,457 Defacement URLs, and 32,520 Malware URLs. Benign URLs denote websites deemed safe for browsing [129], encompassing well-known search engines or social networking platforms. Conversely, Malware URLs introduce malicious software into a user's device upon visiting the URL [115]. URLs categorized as Defacement [6] are crafted by hackers intending to substitute the original content of a hosted website with their own. Phishing URLs [57] are designed to acquire sensitive information like login credentials and financial data illicitly [69]. Sourced from various repositories, including the URL dataset (ISCX-URL-2016), Malware domain blacklist dataset for increasing phishing and malware URLs, Faizan's GitHub repository for augmenting benign URLs, and additional phishing URLs from datasets like Phishtank and PhishStorm, this dataset offers a comprehensive collection for analysis. Table 4.6 presents a small sample of the chosen dataset:

Table 4.6: Example Subset of the « Malicious URLs dataset» [118].

URL	Type
br-icloud.com.bk	phishing
bopsecrets.org/rexroth/cr/.htm	benign
http://www.824555.com/app/member/SportOption.php?uid=guest&langx=gn	malware
http://www.garagepirenne.be/index.php?option=com_content&view=article&id=70&vsig70_0=16	defacement
infinitysw.com/	benign

4.2.1.2 Data Preprocessing

Preprocessing plays a crucial role in tackling various challenges within the dataset, including inconsistency, incompleteness, error detection, and duplicate and null values management. This critical stage encompasses several essential steps, such as feature selection, addressing class imbalances through oversampling or undersampling techniques, performing feature engineering, converting data types, augmenting data, and normalizing and scaling the dataset. Furthermore, data is partitioned into training and testing subsets. All these steps will result in data that is ready for input into the model [16], [71]. That is why the performance of a DL classifier depends on selecting the most effective preprocessing techniques.

4.2.1.3 The deep learning classifiers used

The phase of building and training a DL model for classification involves the planning and construction of a suitable architecture through thorough research in the target domain. It also includes

the definition of model metrics, training and validation of the model using the respective datasets, interpretation of the results, and their comparison with related works [16], [71]. The following items describe the functioning of deep learning classifiers used in this thesis.

1. **Convolutional Neural Networks**
   Convolutional networks, often referred to as convolutional neural networks (CNNs), constitute a specialized type of multilayer neural network. Their architectural layout draws inspiration from the visual cortex of mammals, reflecting the discovered visual processing mechanisms in biological organisms. CNNs, as artificial neural networks, possess the ability to categorize information across a spectrum ranging from rudimentary to intricate [40]. The initial CNN prototype was crafted by LeCun et al., marking a significant milestone in this field [84].

   - **The layers of a CNN**
     The foundational architecture of a Convolutional Neural Network (CNN) consists of a sequence of processing layers, including the convolutional layer, a pooling layer, an activation function, and a fully connected layer. In this section, we provide a brief overview of the commonly encountered layers, which have been utilized in our experimental phase. Figure 4.3 presents a standard architecture of a convolutional neural network [40], [8], [17].

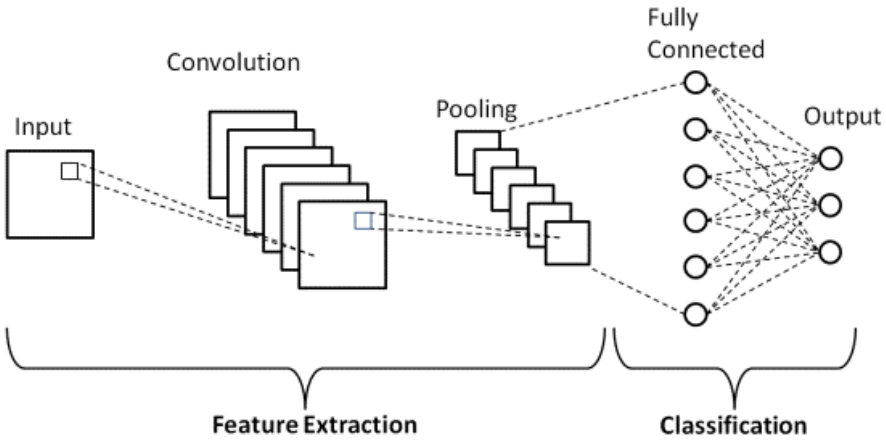


Figure 4.3: Basic architecture of the CNN [17].

     – **The convolutional layer**
       This layer constitutes the fundamental building block of a Convolutional Neural Network, playing a central role in computational tasks. It executes convolutional operations on the input data, employing filters (kernels). The convolution process entails the movement of a kernel or filter within this layer across the receptive fields of the input data, examining the presence of features in the input

data. Through successive iterations, the kernel traverses the entire input data, generating a feature map or convolved feature as a result. Figure 4.4 displays the calculation of the convolution operation [92].

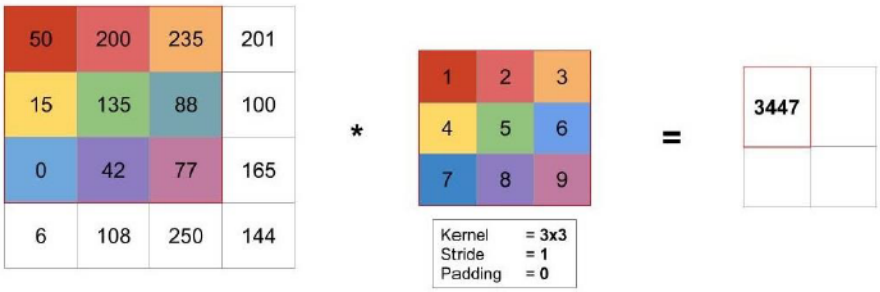


Figure 4.4: The convolution operation [92].

- **Pooling layer** Pooling layers serve the purpose of diminishing the spatial dimensions of the input volume, consequently reducing the complexity and mitigating the risk of overfitting. Common pooling methods encompass max pooling and average pooling. The pooling layer contributes to a reduction in the number of parameters and computations within the network, thereby enhancing overall efficiency. Figure 4.5 demonstrates the process of both max pooling and average pooling operations [111]. In our CNN architecture, we utilized the max pooling layer.
- **Activation function (ReLu)**
  Activation functions are non-linear operations that involve a single numerical input, undergoing specific mathematical transformations. Common activation functions include Rectified Linear Unit (ReLU), sigmoid, and tanh. ReLU designates the real non-linear function defined by: $ReLU(\mu) = \max(0, \mu)$. Applying the Rectified Linear Unit can expedite the convergence of gradient descent. The ReLU correction layer functions by substituting any negative input values with zeros. In essence, it serves as the activation function in the neural network architecture [139].

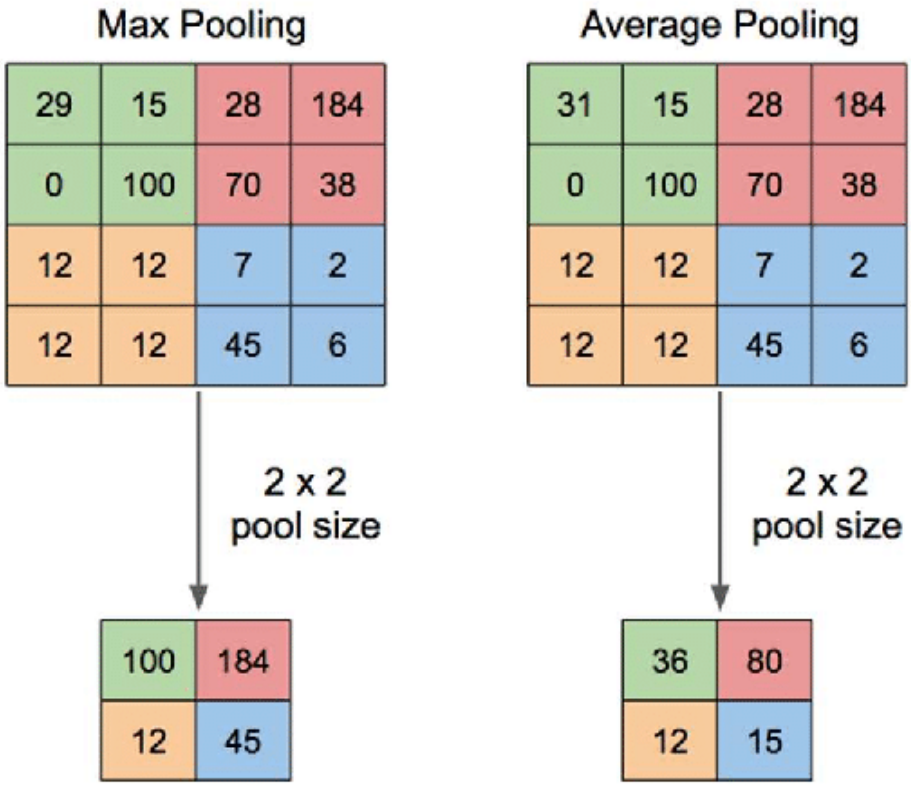


Figure 4.5: Process demonstration of the Pooling operation [111].

– **Fully Connected Layer**

  This layer accepts a vector as input and generates a new vector as output. It achieves this by performing a linear combination of the input values, possibly followed by an activation function. In a classification task, the final fully connected layer is crucial for classifying the network's input. It generates a vector of length N, where N denotes the quantity of classes in the classification issue. Each component of this vector indicates the likelihood of the input being associated with a specific class. Fully connected layers create links between every neuron in one layer and every neuron in the subsequent layer. These layers are commonly employed towards the end of a CNN to generate predictions by leveraging the high-level features acquired from preceding layers. Figure 4.6 shows the structure of the fully connected layer [139].

– **Batch Normalization**

  Batch normalization involves the normalization of a layer's input by adjusting and scaling activations before passing them into the subsequent layer within the network. This process contributes to training stability and can expedite the overall training process [139].

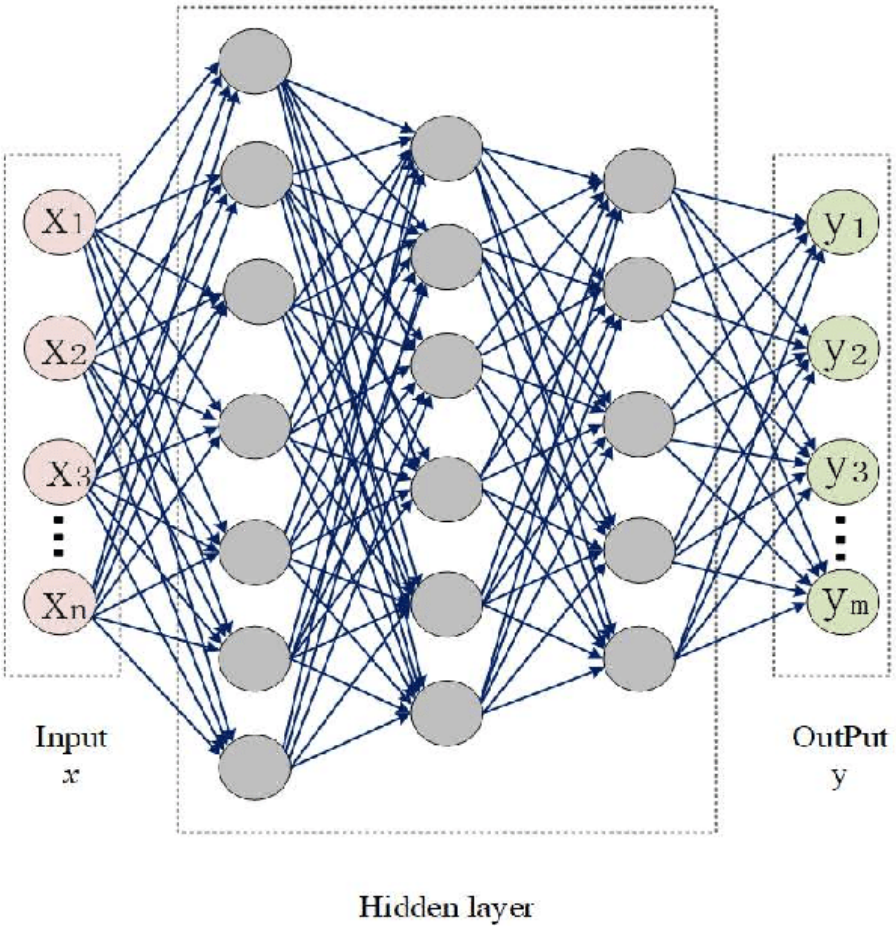


Figure 4.6: The fully connected layer [139].

- **Dropout**
  Dropout serves as a regularization method frequently employed to alleviate overfitting, wherein randomly selected neurons are excluded during training [139].

2. **The Long Short Term Memory (LSTM) functioning**
   The LSTM architecture, introduced by Hochreiter and Schmidhuber in 1997 [64], is a type of recurrent neural network (RNN). Unlike traditional RNNs, LSTM includes a specialized LSTM cell, which plays a crucial role in assessing the significance of information within the algorithm. Over the years, LSTM has found numerous applications in the realms of science and technology [34]. It is a specific type of RNN network that uses a special unit in addition to the standard unit and which aims to deal with the problem of gradient dissipation. This special unit consists of a memory cell that allows the network to store and access information over an extended period.
   Figure 4.7 is an example of a Long Short-Short Memory (LSTM) cell, where “Forget Gate”,

“Input Gate” and “Output Gate” work cooperatively to control the flow of information in an LSTM unit [41].

The input gate (1) determines which information should be updated in the memory cell. It takes into account the current input x_t as well as the previous state of the memory cell h_(t-1) and generates an activation vector that represents the information to be added to the cell c_(t-1). This addition of information results in a mathematical operation carried out between this activation vector and the previous state c_(t-1) [41].

The forget gate (2) allows the LSTM to delete obsolete information from the memory cell. It uses both the current input and the previous state to generate an activation vector that determines what information should be forgotten. This also results in a mathematical operation carried out between this second activation vector and the previous state c_(t-1). It is from the two previous operations on c_(t-1) that we obtain the current state c_t [41].

Finally, the output gate (3) determines the output of the LSTM at a given time. It uses the current input x_t and the current state of the memory cell c_t to generate an activation vector that represents the output of the LSTM. This is how we obtain h_t [41].

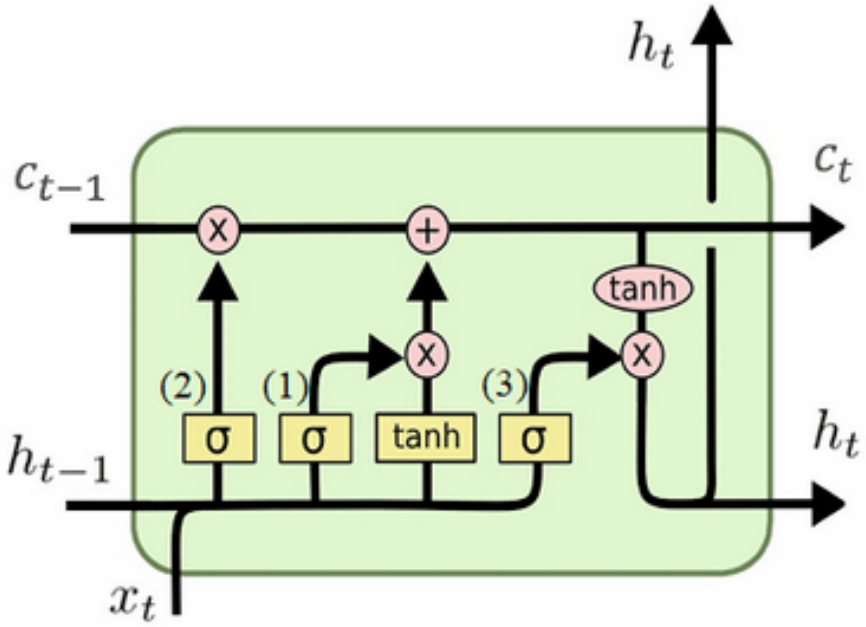


Figure 4.7: The fundamental framework of an LSTM unit.[41].

4.2.1.4 Model evaluation & the performance metrics used

In this phase, the model undergoes testing with data that has not been part of its training process. This evaluation metric enables us to gauge the model's potential performance with previously unseen data, offering insights into how the model might operate in real-world scenarios. This is vital to ensure that the model's inferences are both efficient and valuable [16], [71].
The commonly used performance measures for deep learning-based models in the field of phishing detection are as follows:

1. **The confusion matrix**
   The confusion matrix serves as a condensed representation of the classification model's predicted outcomes, offering an assessment of its performance. It is constructed by aggregating the counts of correctly and incorrectly classified predictions for each class [61], [107], [104], [49]. Figure 4.8 illustrates the confusion matrix applied in our study of phishing classification.

		Actual Class	
		Phishing "1"	**Legitimate "0"**
Predicted Class	**Phishing "1"**	**TP**	**FP**
	Legitimate"0"	**FN**	**TN**

Figure 4.8: The used confusion matrix [107].

   Below are the explanations of True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN): [107]:
   TP: The count of accurately identified phishing URLs.
   TN: The tally of correctly identified legitimate URLs.
   FP: The total of legitimate URLs is incorrectly labeled as phishing URLs.
   FN: The sum of phishing URLs wrongly identified as legitimate URLs.

2. **Accuracy**
   Accuracy refers to the proportion of accurate predictions compared to the total number of

predictions. It is computed using the equation 4.1 [107]:

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

3. **Precision**
   It suggests the assessment of accurately identified positive URLs TP out of the total URLs predicted as positive (TP+FP), and it is computed using equation 4.2 [107]:

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

4. **Recall**
   Recall represents the proportion of correctly predicted positive URLs TP out of all actual positive URLs (TP+FN), and it is determined using equation 4.3 [107]:

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

5. **F1-score**
   This metric gauges a model's accuracy on a dataset, specifically designed for assessing binary classification systems that categorize examples as either "negative" or "positive". It's computed using the equation 4.4 [107]:

$$F1-Score = 2 * \frac{(Precision * Recall)}{(Precision + Recall)} \tag{4.4}$$

6. **False-positive rate (FPR)**
   It measures the rate of false alarms, or how often the model incorrectly labels benign URLs as malicious. It is calculated as the following equation (4.5) [109]:

$$FPR = \frac{FP}{(FP+TN)} \tag{4.5}$$

4.2.1.5 Optimization and parameters tuning

The DL model's performance depends on several parameters. Enhancing the model's training involves the manual tuning of specific parameters like epochs, batch size, and learning rate. This fine-tuning process aims to identify the optimal parameter configuration that yields the highest

results, and it is commonly referred to as hyperparameter tuning. Hyperparameters are distinct from parameters in that their values cannot be directly inferred from the data. Unlike parameters, there is no straightforward method for determining their optimal values. Hyperparameters are being tuned when refining a deep learning algorithm, and they may need to be manually defined and then fine-tuned through iterative experimentation [16], [71].

In our initial contribution 4.3, we conduct a series of experiments to fine-tune three critical parameters: batch size, the number of epochs, and the learning rate.
Batch size denotes the total number of training examples processed in each batch, while an epoch signifies one complete pass of the entire dataset through the neural network, both forward and backward.
The learning rate, on the other hand, governs the speed at which the neural network model adapts to the problem at hand. Identifying the ideal learning rate for a specific model on a particular dataset cannot be achieved through analytical means. Instead, we must determine an effective learning rate through the process of hyperparameter tuning [16], [71].
The utilization of an improper learning rate can lead to overfitting in the model, wherein the model demonstrates exceptional performance on the training data than the test data. To mitigate this issue, the early stopping regularization technique is commonly employed in deep learning due to its effectiveness and simplicity. This method involves terminating the training process once the model performance stops improving and the error begins to rise on the validation set [71, 25].

4.3 First Contribution: A deep learning approach to detect phishing websites using CNN

In this section, we propose a phishing website detection system using three types of features: URL-based features, content-based features, and third-party services-based features. The proposed detection technique is based on deep learning technology, specifically testing both 1D CNN and 2D CNN models. The use of the 2D CNN model in this context is novel, as it is commonly employed for image classification, and we apply it for the first time to this type of phishing dataset. Additionally, we conduct experiments to optimize parameters for each CNN model, aiming to assess the impact of the parameter tuning process on the deep learning model's performance. The primary objective is to establish an effective phishing detection mechanism to enhance the cybersecurity of users on the web. This work was the subject of our article "A deep learning approach to detect phishing websites using CNN for privacy protection", IDT journal. 2023. [107]. Section 4.3.1 presents a description of the proposed approach, defining each step within it. Then, Section 4.3.2 delves into the implementation and evaluation of the performance of this approach. Finally, Section 4.3.3 offers a comparison of this approach with existing literature.

4.3.1 Description of the proposed anti-phishing mechanism

The proposed approach for detecting phishing websites is a combination of several main processes. Figure 4.9 outlines the general framework of our proposed system. The first process in this framework is the collection of a dataset that involves features from both phishing and benign URLs. This dataset was gathered from sources such as Alexa and PhishTank, and it is in CSV format indexed in the public Kaggle and mendeley repositories. It comprises 87 features with their corresponding URLs, totaling 11,430 data points, including 5,715 phishing samples and 5,715 legitimate samples. The input data includes three types of features: URL-based features, content-based features, and third-party services-based features. Subsequently, these features undergo the second process, which is pre-processing. This involves testing and removing null cases, shuffling the two classes, splitting the data into train and test, and adapting the type and format of the samples to meet the input shape required by the model, among other steps. This process results in input data that is ready to be introduced into the deep learning classifier.
After that, in the classification model step, 1D CNN and 2D CNN models are applied to the training dataset, producing an output indicating whether the website is legitimate or phishing. The novelty lies in the application of the 2D CNN with a time series dataset. Finally, the results are evaluated against the test set using various evaluation metrics such as accuracy, precision, recall, and F1-score. The following subsections present these steps in detail.

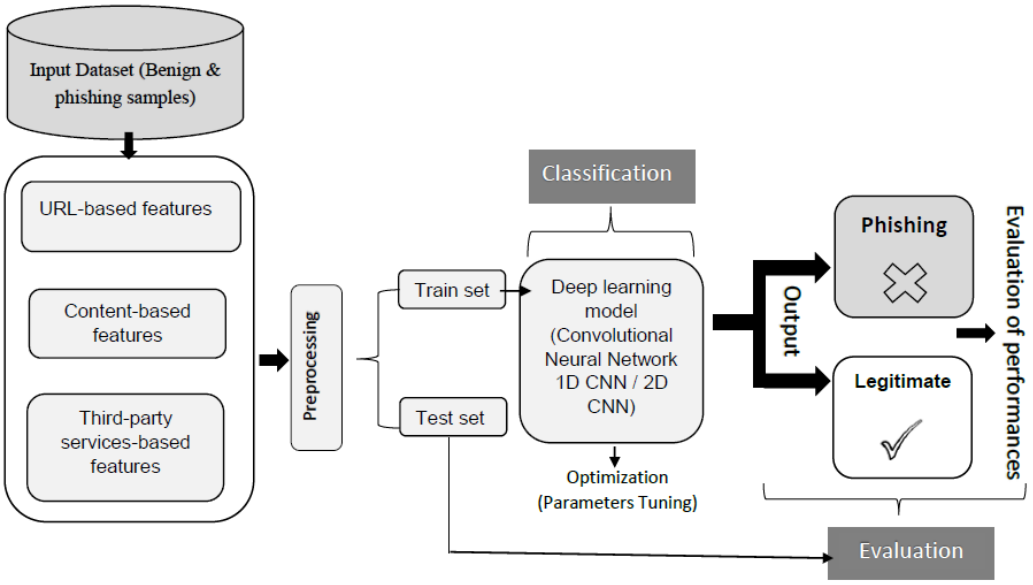


Figure 4.9: General framework of the 1st proposed approach.

4.3.1.1 Input dataset

For this study, we utilized the "Web Page Phishing Detection" dataset [60] sourced from Mendeley Data to formulate and evaluate our proposed method. This dataset was selected due to its extensive size and comprehensive features, facilitating a deeper analysis of website characteristics. Additionally, it encompasses three types of features, enhancing the robustness of detection. As delineated in item 1 of subsection 4.2.1.1, the dataset comprises 11,430 URLs, evenly distributed with 50% (5715) labeled as "legitimate" and 50% (5715) as "phishing", ensuring balance in the dataset.

4.3.1.2 Preprocessing steps used

As explained in section 4.2.1.2, The initial phase in the deep learning process involves preprocessing the dataset to ensure its readiness for integration into the classification model. The steps carried out in this phase include [107]:

- **Data cleaning:** it involves removing duplicate URLs and checking for missing values using the function ‘dataframe.isnull().sum()‘, followed by their deletion if any are found.

- **Balancing and shuffling the datasets:** this step involve using the function ‘data[’status’].value_counts()‘ to count the samples of each class (phishing and legitimate). If the two classes are not equal, they must be balanced. In this study case, the dataset contains 5715 phishing samples and 5715 legitimate samples. Subsequently, the dataset is shuffled using the ‘append()‘ function.

- **Digitizing the values in the data frame:** it involves converting non-numeric data types to numeric types using the ‘astype(int)‘ function. This ensures that the data is adapted to the input shape of the CONV layer. Additionaly, the "status" column is assigned binary values, where the phishing class is represented by 1, and the legitimate class is denoted by 0.

- **Defining the inputs and outputs:** In this context, we have defined the input data as the features, and the output is identified by the status index (1 for phishing and 0 for legitimate).

- **Adapting the input dataset to meet the input shape requirements of the deep learning model (1D CNN/2D CNN):** it involves using the following functions: StandardScaler(), which facilitates standardized features by removing the mean and scaling to unit variance. Next, the data frame is reshaped into an array using the ‘np.array()‘ function. The size of the input data is then adjusted using the ‘reshape()‘ function (2 dimensions for 1D CNN and 3 dimensions for 2D CNN).

- **Splitting the dataset into training and testing sets:** involved dividing the dataset, which consists of 11,430 rows, using the scikit-learn Python library and the "train_test_split" function. We allocated proportions of 80% for training and 20% for testing, resulting in 9144 samples for the training dataset and 2286 samples for the testing dataset.

4.3.1.3 Classification model

The detection system comprises a Convolutional Neural Network, a type of deep, forward-feeding artificial neural network. Its architecture encompasses an input layer, four layers of 1D convolution, three layers of 1D pooling, a flattened layer, two dense layers, and an output layer delineating two categories (phishing or legitimate). A comprehensive breakdown of the 1D CNN classification model and its constituent layers is illustrated in Figure 4.10.
At first, the inputs pass through two 1D convolution layers, employing ReLU as the activation function with 32 and 64 filters, respectively. The output of these convolution layers comprises feature maps, which are then subjected to a pooling layer to reduce the number of parameters and calculations in the network. This process is repeated for subsequent Conv1D (64 filters), max pool 1D, Conv 1D (128 filters), and max pool 1D layers. Following this, the flatten layer is activated, and its output is directed to a dense layer featuring 64 neurons. The dense layer also utilizes ReLU as the activation function, and its output is further directed to another dense layer with a single neuron and a sigmoid activation function. Dropouts are incorporated after each pooling layer and following the initial dense layer. Additionally, the rectified linear unit (ReLU) activation function is applied after each convolutional layer. The Adam optimizer is employed, and binary cross-entropy serves as the loss function.
We replicated the same scenario using the 2D CNN (Convolutional Neural Network), adjusting the data dimensions and the parameters of each layer to meet the requirements of this specific model. Figure 4.11 displays a detailed architecture of the 2D CNN model.

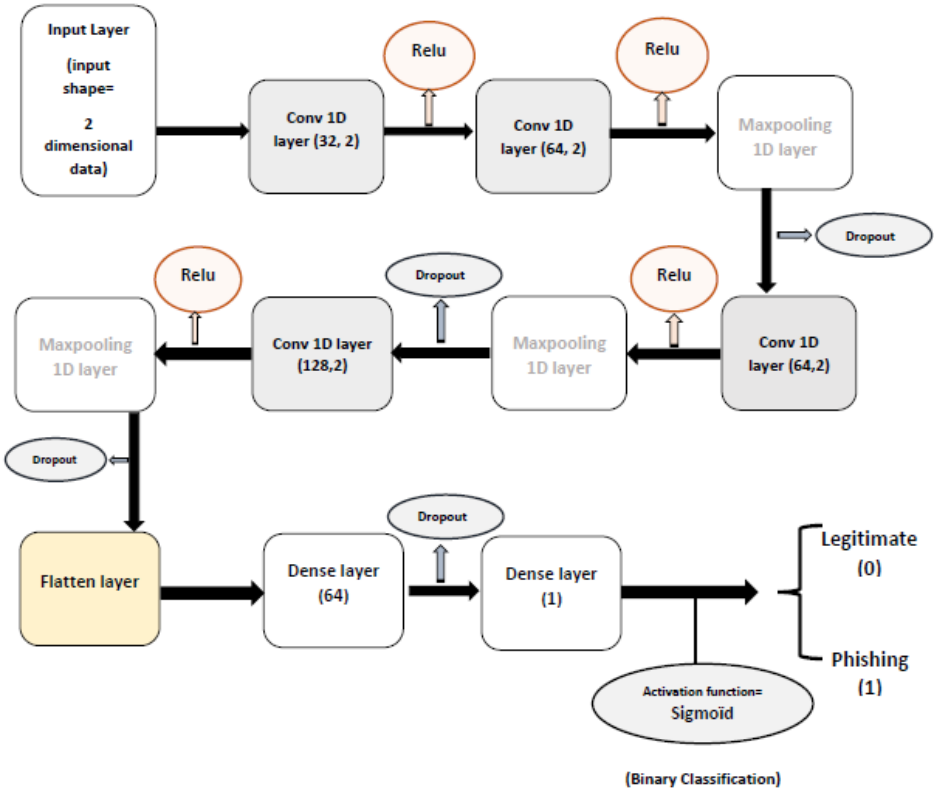


Figure 4.10: The architecture of the 1D CNN classifier.

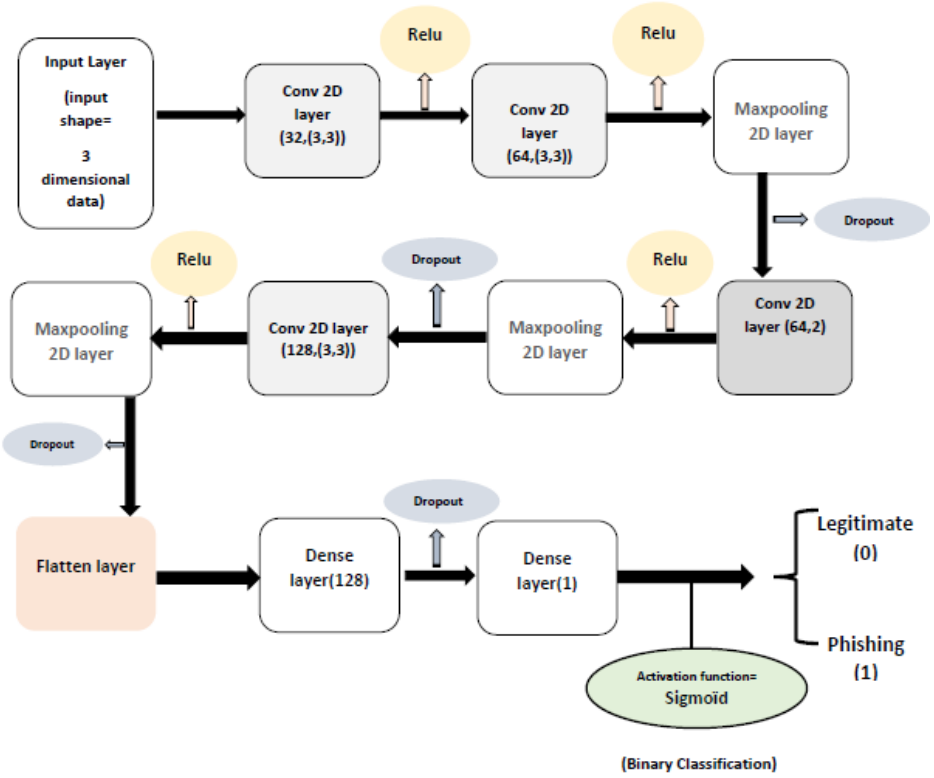


Figure 4.11: The architecture of the 2D CNN classifier.

4.3.2 Experimentation and Evaluation

The two aforementioned models, 1D CNN and 2D CNN, are compiled independently. The training dataset, comprising 80% as mentioned earlier, is utilized for training, while the testing dataset, constituting 20%, is employed for testing. Then, each model is evaluated using the performance measures mentioned in section 4.2.1.4.

4.3.2.1 Environment of Work and Utilized Tools

We implemented the proposed approach in the following environments:

- OS: Windows 10 (64-bit)
- Jupyter notebook within the Anaconda 3 environment
- Utilization of Python (Version 3.9) for programming

- Employing TensorFlow and Keras libraries, which are open-source machine learning frameworks built upon Python.

- Visualization of the model's accuracy and loss graphs is facilitated through the use of the matplotlib.pyplot and learning_curve functions.

4.3.2.2 1D CNN Model

The performance of the 1D CNN model is influenced by various parameters, necessitating the selection of a parameter set that maximizes performance accuracy, as mentioned in subsection 4.2.1.5. This process is known as parameter tuning [45]. The list of parameters defined during the design of the 1D CNN model is presented in Table 4.7 where parameters such as epochs, batch size, and learning rate are systematically tested through a series of experiments, adjusting their values to obtain optimized settings [107]. The Adam optimizer algorithm is employed, as it has been demonstrated in numerous studies to be among the most effective optimization techniques [5].

Table 4.7: Parameters used during the implementation of the 1D CNN Model [107].

Parameter	Value
Activation function	Sigmoid for binary classification
Optimizer algorithm	Adam
Loss function	Binary cross-entropy
Dropout rate	0.5
Learning rate	Table 4.8
Batch size	Table 4.9
Number of epochs	Table 4.10

- **Experiments and Analysis of results**

1. **Experiment 1: Fine-tuning the Learning Rate Setting**
   The learning rate is a parameter that regulates the pace at which the neural network model learns a given problem. More details are given in section 4.2.1.5. In this experiment, we varied the learning rate within the range of 0.0001 to 0.05 to identify the optimal value for this study CNN architecture, aiming for the highest detection accuracy. The learning rate interacts with numerous other optimization process parameters, including the number of epochs and batch size, and these interactions may exhibit nonlinearity [26, 27]. The initial values for batch size and the number of epochs were arbitrarily set to 64 and 50, respectively. Table 6 presents the results obtained from Experiment 1. Notably, we observed that an optimal accuracy value (96.29%) is achieved when the learning rate is set to 0.001. Conversely, setting the learning rate to 0.05 resulted in a sudden accuracy drop to 50%. A higher learning rate, as in the case of 0.05 in this experiment, tends to provide satisfactory performance only in the initial stages. However, over time, the model's performance deteriorates, leading to premature cessation before achieving optimality. Moreover, it can prompt the model to converge too swiftly to a suboptimal solution [26, 27].

Table 4.8: Evaluation of 1D CNN Model Performance Across Distinct Learning Rates [107]

Learning rate	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	Time(s)
0.0001	95.31	95	95	95	150
0.0005	96.27	95.47	96.75	96.10	151
0.001	**96.29**	**96.16**	**96.41**	**96.28**	**150**
0.005	95.97	95.56	96.40	95.98	150
0.01	95.05	96.81	93.17	94.95	352
0.05	50	50	50	66.67	160

2. **Experiment 2: Fine-tuning the Batch Size Setting**
   As mentioned in section 4.2.1.5, the batch size represents the total number of training examples processed in each batch [16, 71]. In this experiment, using the learning rate of 0.001 obtained from the previous experiment, the epoch is still randomly set to 50. We tested batch size values ranging from 16 to 64, and their corresponding detection measures were assessed. As illustrated in Table 4.9, a batch size of 32 yields the highest accuracy value of 96.41%.

Table 4.9: Evaluation of 1D CNN Model Performance Across Different Batch size values [107]

Batch size	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	Time(s)
16	95.92	94.71	97.28	95.98	250
32	**96.41**	**96.09**	**96.76**	**96.43**	**200**
64	96.28	96.15	96.42	96.27	150

3. **Experiment 3: Fine-tuning the epochs setting**
   An epoch represents one complete pass of the entire dataset through the neural network, both forwards and backward [16, 71]. With the learning rate set to 0.001 and a batch size of 32 obtained from the previous experiments, we increased the training epoch from 50 to 300 to determine its optimized value. As indicated in Table 4.10, the model achieves its highest accuracy value (96.76%) at 250 epochs.
   At this juncture, the optimal set of parameters that yields the best detection accuracy has been identified. The parameters providing the best performance for the 1D CNN model in this study are (learning rate = 0.001, batch size = 32, epochs = 250).

Table 4.10: Evaluation of 1D CNN Model Performance Across Different Epochs Values [107]

Epochs	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	Time(s)
50	96.40	96.08	96.75	96.42	200
100	96.32	96.15	96.51	96.32	705
150	96.58	96.57	96.57	96.57	900
200	96.32	96.58	96.84	96.71	400
250	**96.76**	**96.75**	**96.50**	**96.62**	**500**
300	96.53	96.32	96.57	96.45	600

The loss and accuracy graphs for the 1D CNN model across different numbers of epochs (ranging from 50 to 300) with a learning rate of 0.001 and a batch size of 32 during both training and validation are depicted in Figure Figure 4.12. Upon analyzing Figure 4.12, it becomes apparent that the graph at 250 epochs yields the highest accuracy value, indicating proficient model performance without overfitting. The model's accuracy on validation steadily increases with the growth of epochs and begins to stabilize after 50 epochs, ultimately reaching an accuracy value of 96.76% at epoch 250, which is commendable. Similarly, the model's loss in validation gradually decreases, reaching approximately 12% at epoch 250, underscoring the effectiveness of the 1D CNN model in phishing detection.

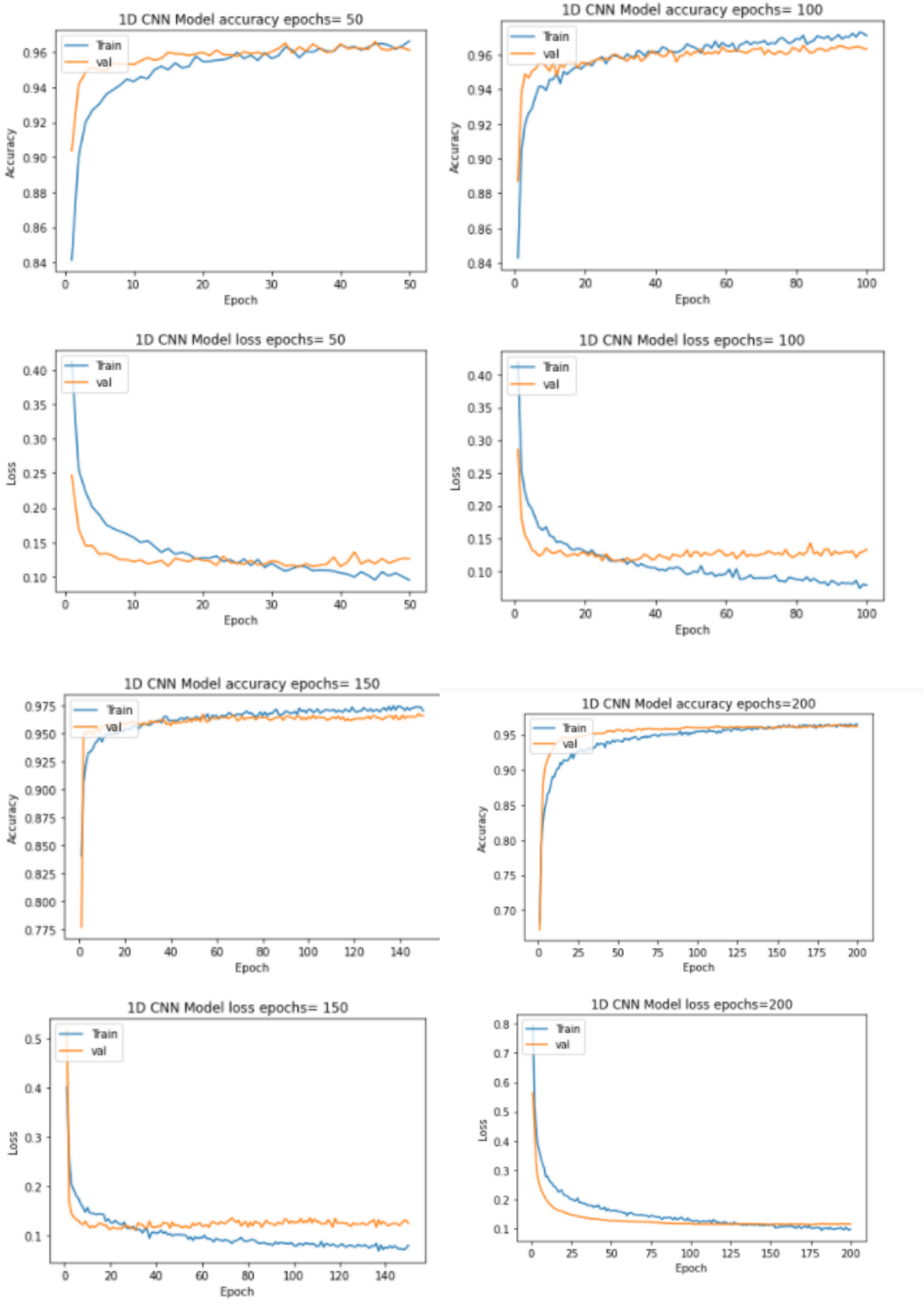


Figure 4.12: Graphs depicting the accuracy and loss trends of the 1D CNN Model across different numbers of epochs (ranging from 50 to 300), utilizing a learning rate of 0.001 and a batch size of 32 [107].

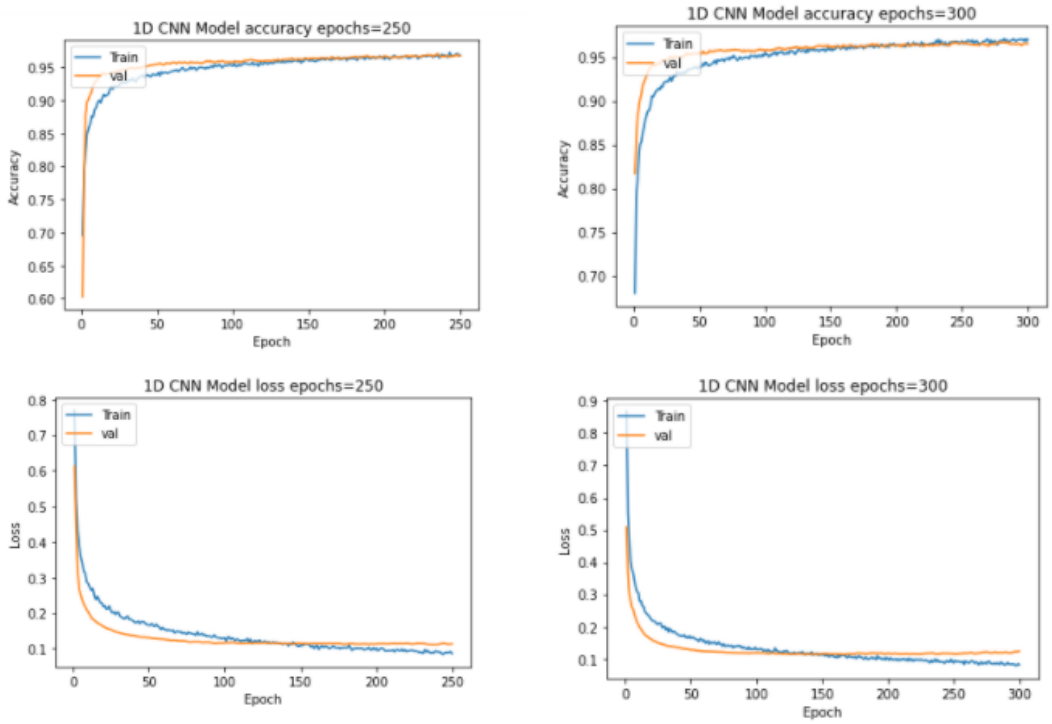


Figure 4.12: Continued.

4.3.2.3 2D CNN model

The parameters specified during the design of the 2D CNN model are outlined in Table 4.11. Furthermore, Table 4.12 displays the results achieved by the 2D CNN model for varying numbers of epochs.

Table 4.11: Parameters Utilized in Configuring the 2D CNN Model [107].

Parameter	Value
Activation function	Sigmoid for binary classification
Optimizer algorithm	Adam
Loss function	Binary cross-entropy
Learning rate	0.001
Dropout rate	0.5
Batch size	64
Number of epochs	4.12

Table 4.12: Evaluation of 2D CNN Model Performance Across Different Epochs Values [107]

Epochs	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	Time(s)
50	95.75	94.82	96.87	95.84	226
80	**95.84**	**94.77**	**97.14**	**95.94**	**320**
100	95.48	95.26	95.84	95.55	400
100 (with early stopping) 4.2.1.5	95.49	95.67	95.40	95.53	8

Figure 4.13 illustrates the accuracy and loss graphs of the 2D CNN model for various numbers of epochs (50, 80, 100) during both training and validation.
Upon analyzing Figure 4.13, in the graph where epoch = 50, it is evident that the model performs well with no signs of overfitting. The accuracy on validation steadily increases with the growth of epochs, approximately stabilizing after the 15th epoch and reaching a value of 95.76% at epoch 50, which is commendable. However, as the number of epochs increases to 80 and 100, the graph shows instability. The model performs better on training than on validation, and the validation loss is higher than the training loss, indicating the presence of overfitting. As depicted in Table 10, the validation accuracy attains a 95.84% value at 80 epochs and a 95.49% value at 100 epochs.
To address overfitting, the early stopping technique using the callback function is employed. However, as displayed in the graph (epochs = 100 using early stopping), the model accuracy stops improving after the 2nd epoch due to overfitting.

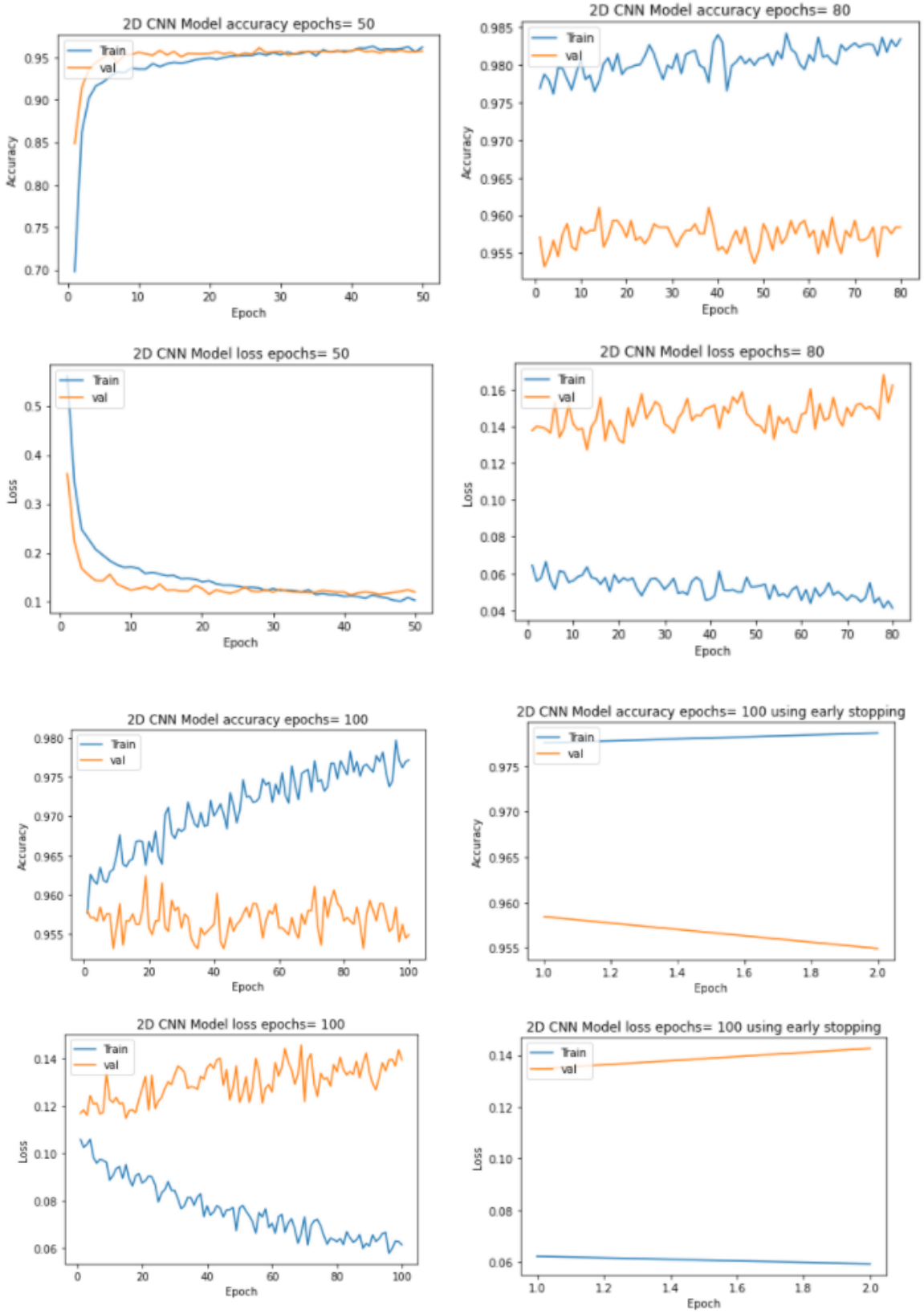


Figure 4.13: Graphs depicting Accuracy and Loss for the 2D CNN Model across Different Epoch Counts (50, 80, and (100 pre and post early stopping))[107].

4.3.3 Comparison with literature

Table 4.13 displays the outstanding results achieved by the proposed approach for both 1D CNN and 2D CNN, as indicated by the performance measures provided in section 4.2.1.4. These results are further compared with those obtained by other relevant works mentioned in the state-of-the-art chapter 3. The comparison is specifically focused on works that used the same dataset as ours and also employed CNN as a classification model.

Figure 4.14 depicts a comparison of the proposed approach with the related studies in terms of accuracy.

As depicted in table 4.13, the 1D CNN model attained an accuracy of 96.76%, precision of 96.75%, recall of 96.50%, and an F1-score of 96.62%. These metrics indicate the strong performance of the 1D CNN model in detecting phishing websites, surpassing the accuracy values reported by authors in [7], [136], [82], who also employed 1D CNN and achieved accuracy rates of 95.02%, 92.05%, and 93%, respectively. Moreover, our model outperformed other studies employing the same dataset in terms of accuracy and various performance metrics, except for the Random Forest model in reference [127], which achieved a 97% accuracy, close to our results. We chose to compare our work with machine learning-based approaches because the dataset used in our study has primarily been utilized by such works in the literature, which is an advantage for our research. Notably, this marks the first instance of employing deep learning techniques on this dataset. Although deep learning techniques are recognized for their prolonged training times, our proposed approach's 1D CNN model exhibits efficient training, completing in 500 seconds, which is notably fast compared to the 5281.81 seconds taken by the CNN model in [7], considered relatively lengthy.

Based on the related works cited in this thesis, this marks the initial application of 2D CNN in the field of phishing detection, a model typically employed in image classification tasks. The 2D CNN model in our proposed approach achieved an accuracy value of 95.84%, surpassing the accuracy values reported in [88], [7], [136], [82]. However, upon closer examination of the model accuracy and loss graphs with an increase in epochs, it exhibited suboptimal performance and indicated overfitting. This could be attributed to the nature of the data and the input shape size, as we needed to adjust the dimensions of the input data to align with the requirements of the 2D CNN model. Despite this, the accuracy remains commendable.

These results lead us to infer that 1D CNN is better suited for phishing detection, especially when handling numerical datasets. On the other hand, 2D CNN might be more fitting for image detection tasks, however, given that this is the first application of 2D CNN with this type of dataset, it has demonstrated effective performance in phishing detection.

Table 4.13: Comparison of Results Achieved by the Proposed Approach and State-of-the-Art Techniques.

	Our 1D CNN model	**Our 2D CNN model**	**[127]**	**[88]**	**[112]**	**[7]**	**[136]**	**[82]**
The model used	1D CNN	2D CNN	Decision Tree, Random Forest, KNN, Gaussian Naïve Bayes, and XGBoost	Gradient Boosting ensemble learning	XGBoost, Linear Learner, and k-Nearest Neighbor (k-NN)	1D CNN	1D CNN	1D CNN
Accuracy (%)	**96.76**	95.84	DT =93.57, RF=97, KNN=83.33, GNB=74.02, XG-Boost=94.79,	95.36	XGBoost= 96.41, Linear Learner (LL)=94.43, KNN=83.73	95.02	92.05	93
Precision (%)	96.75	94.77	/	96.29	XGBoost== 97.09, LL=95.26, KNN=91.15	92.35	99.44	92
Recall (%)	96.50	97.14	/	94.24	XGBoost= 95.80, LL=93.67, KNN =75.22	98.09	84.57 /	
F1-score (%)	96.62	95.94	DT= 93.64, RF=96.99, KNN=82.22, GNB=70.30, XG-Boost=94.73	95.26	XGBoost= 96.44, LL=94.46, KNN=82.42	95.13	91.41	91
Training time (s)	500	320	/	/	/	5281.81	/	/

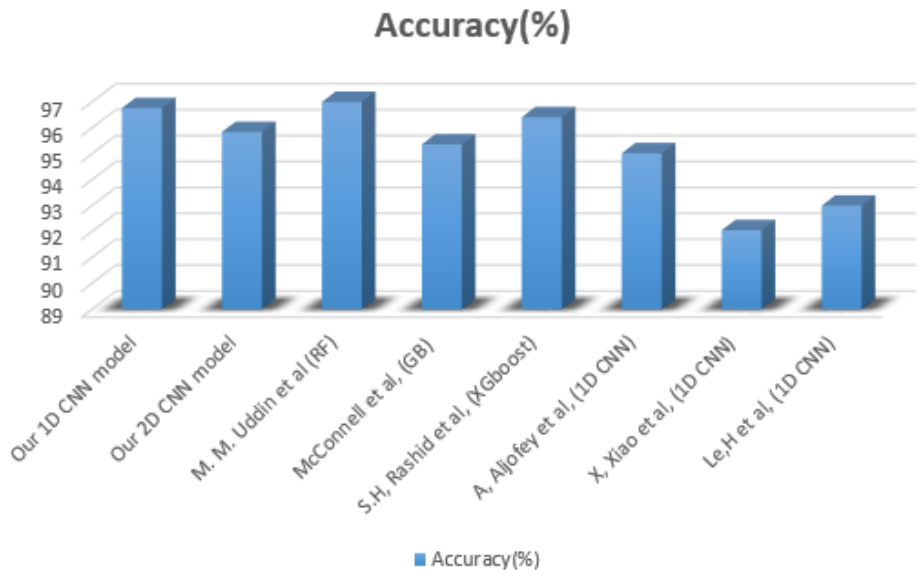


Figure 4.14: Comparison with related studies.

4.4 Second Contribution: Detection of Phishing Websites using the Permutation Importance Method and DL-ML Models

This contribution presents an intelligent approach to identifying phishing URLs using multiple classifiers. The method employs the permutation importance method PIM to select crucial URL features and the Smote-Tomek link method to address dataset imbalances. Following preprocessing, these selected features are fed into The XGBoost and four deep learning classifiers: CNN, LSTM, and two hybrid models (CNN-LSTM & LSTM-CNN) for classification, aiming to identify the most effective model for phishing detection. Experimental findings demonstrate the superior performance of our proposed mechanism compared to existing studies employing the same dataset. Accuracy ranges between 94.12% to 97.82% was achieved with feature selection and dataset balancing. Finally, to improve accessibility for web users, our phishing detection mechanism has been implemented as a web application. This work was the subject of our article "A Deep Learning Mechanism to Detect Phishing URLs using the Permutation Importance Method and SMOTE-Tomek Link", Journal of Supercomputing. Springer. 2024 [108].
Subsection 4.4.1 describes the proposed mechanism, detailing each step. Subsection 4.4.2 offers the implementation and evaluation, with Subsubsection 4.4.2.1 providing an analysis of the achieved results. Subsubsection 4.4.2.2 delves into the details of the web application implementation, while Subsubsection 4.4.2.3 presents a comprehensive comparison with the existing literature.

4.4.1 Method Description

In this section, we present a comprehensive overview of the proposed phishing detection mechanism. The mechanism comprises several steps. To commence, the dataset employed consists of two variants of the "Mendeley phishing websites dataset", namely D1 and D2 [133] (this dataset is detailed in sect2). To augment the performance of our approach, we applied several operations to the data in the preprocessing phase. These operations encompassed data cleaning, entailing the removal of duplicate and null samples, dataset balancing, feature selection using the permutation importance method, which identifies the K top most relevant features of each dataset, and dataset splitting.
The dataset balancing operation adopted the SMOTE Tomek link technique, which combines oversampling and undersampling methods to address the class imbalance issue. The subsequent stage involved embedding the treated dataset into the classification model, where the XGBoost and four distinct deep-learning classifiers were trained: CNN, LSTM, and two hybrid classifiers (CNN-LSTM and LSTM-CNN). These classifiers generated a classification output, effectively distinguishing between phishing and legitimate websites.
Following this, the proposed mechanism's performance was evaluated using various metrics, ensuring a comprehensive assessment of its efficacy. Finally, our phishing detection mechanism was implemented as a web application to provide a final product to end users and facilitate the usability

of the model for web users. The web application will be detailed in the experimentation section. Figure 4.15 illustrates a detailed conceptual framework of the proposed method [108].

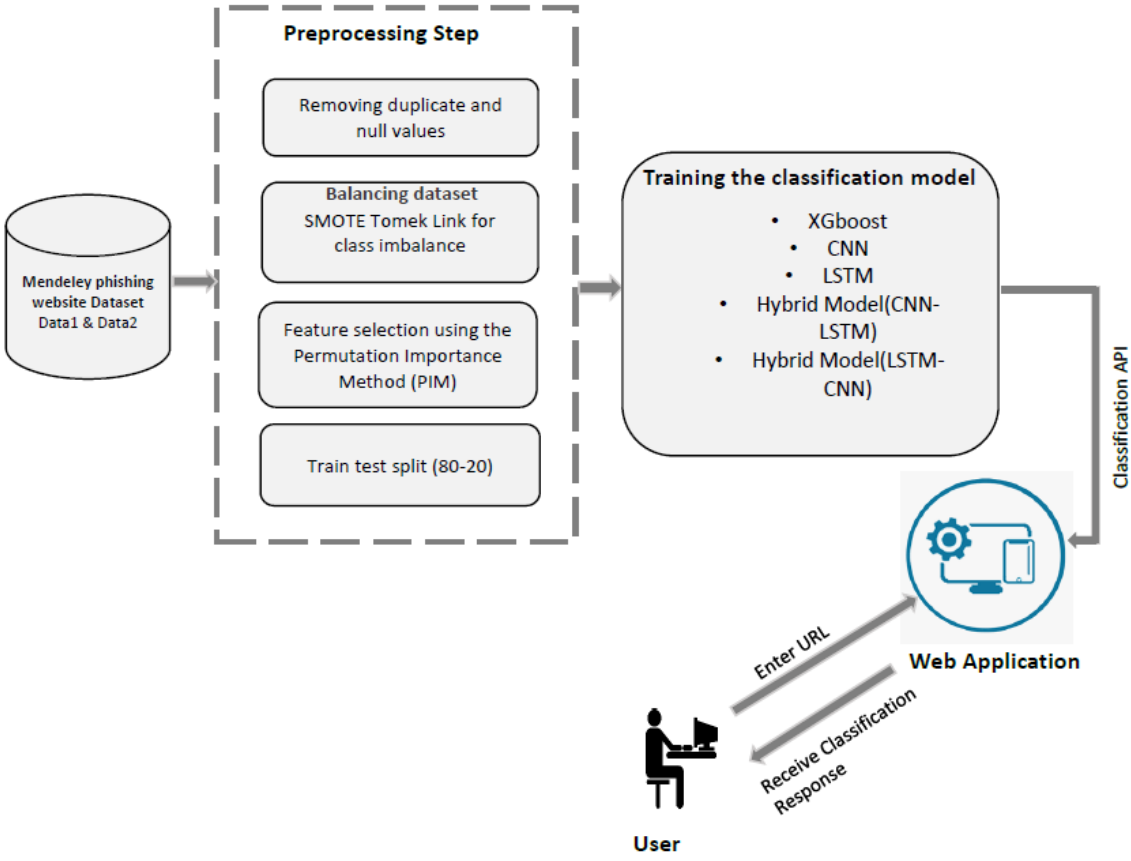


Figure 4.15: Conceptual framework of the 2nd proposed approach.

4.4.1.1 Preprocessing phase

Since the datasets we're working with are already devoid of null and duplicate values, our preprocessing stage will focus on applying balancing techniques, selecting features, and partitioning the data for training and testing, as detailed in the subsequent sections.

- **Data balancing technique (The SMOTE-Tomek Link)**
  The two variations of datasets (referred to as D1 and D2) utilized in this study (sect 2) encountered an issue with the imbalanced class distribution. To address this challenge, we employed a technique called dataset balancing to establish a balanced and dependable dataset. While various balancing methods exist in related literature, we chose the Smote-Tomek link approach as it appeared to be the most suitable solution for preserving significant instances.

In addition, the rationale for selecting SMOTE Tomek links to balance data lies in its implementation process. Unlike other methods like Synthetic Minority Over-sampling Technique (SMOTE) or Synthetic Minority Over-sampling Technique plus Edited Nearest Neighbors (SMOTEENN), which balance data only within the training or test sets, SMOTE Tomek operates on the entire dataset X and Y before their division into train and test sets. This comprehensive approach enhances the reliability of results, minimizing the risk of overfitting or underfitting. Furthermore, this balancing method has not been extensively investigated in conjunction with deep learning models.

Fundamentally, dataset balancing involves two main strategies: oversampling and undersampling. Oversampling entails duplicating random samples from the minority class without introducing new information to the dataset. Conversely, undersampling involves removing random samples from the majority class, potentially leading to the loss of valuable data in the process [131].

The SMOTE technique is frequently employed to tackle class imbalance through oversampling. It generates additional instances to augment the category with fewer samples. This method relies on the k-nearest neighbor algorithm to identify the nearest data points to a randomly selected sample from the minority class. By combining this sample with its k-nearest neighbors, synthetic data samples are created. SMOTE operates by identifying instances close to each other in the feature space, establishing connections between them, and then generating new samples along these connections. Several variants of SMOTE have been proposed in various research studies. In this study, we employed the SMOTE-Tomek link technique, which merges the SMOTE algorithm for oversampling with the Tomek link algorithm for undersampling. In the undersampling phase, instances with the shortest Euclidean distance are eliminated from the predominant class. The Tomek Links procedure involves selecting random instances from the predominant class and verifying if their nearest neighbor belongs to the minority class, signaling a Tomek Link. Any identified Tomek Links are then eliminated. The process of the Smote-Tomek link is described in Algorithm 1 [35], [91], [131].

In dataset D1, the legitimate class is initially identified as the minority class, containing 27,998 samples, while phishing constitutes the majority class with 30,647 samples. Following the application of the SMOTE Tomek link balancing method, both classes are balanced at 29,200 samples each for the two classes. Conversely, in dataset D2, the legitimate class is recognized as the majority class, comprising 58,000 samples, while phishing is the minority class with 30,647 samples. Upon applying the SMOTE-Tomek link balancing method, the two classes are balanced at 56,605 samples each [108].

Algorithm 1 Algorithmic representation of the Smote-Tomek link method [91] [108].

Input: Unbalanced data(D)
Output: Resampled data(D')

1: Select x_i randomly from the minority class
2: Identify k-nearest neighbor of x_i: ψx_i
3: **Generate** $x_{new} = x_i + (\hat{x}i - x_i) \times \delta$
4: **if** !(balancing ratio satisfied): **then** Go to 1
5: **else**: Eliminate Majority sample using Tomek. TOMEK(l,m) { l is the nearest neighbor of m. m is the nearest neighbor of l. l and m belong to distinct classes.}
end if
6: return D'

- **Features Selection using the Permutation Importance Method (PIM)**
  Feature selection entails removing redundant and insignificant features to obtain an optimal subset. Consequently, the primary goal of feature selection algorithms is to reduce the size of the original dataset, thereby decreasing model complexity and processing time, and ultimately enhancing performance [94]. Numerous feature selection methods have been proposed, many of which face challenges such as high computational costs and complexity of application [3]. The approach employed to select the relevant features in this study is the permutation importance method, chosen for its ability to address these limitations. Permutation importance serves as an explanatory technique to assess the significance of features by analyzing their influence on the predictions made by a trained machine learning model. It gauges the importance of a feature to the model's predictions by evaluating the change in the model's error when the value of that feature is permuted. Essentially, the higher the resulting error, the more critical the feature is deemed for accurate predictions [134].
  The equation to determine the Permutation Importance (PI) score of a feature, representing the impact of that feature on the model's performance, is given by the following formula: (eq 4.6) [108]:

$$PIi = \text{Performance Measure (Initial features)} - \text{Performance Measure (Permutated feature)} \tag{4.6}$$

  Where:

  - PIi: represents the Permutation Importance score for feature i.
  - Performance Measure (Initial features): denotes the model's performance metric (such as accuracy or F1 score) when trained on the initial dataset.
  - Performance Measure (Permutated feature): indicates the model's performance metric when trained on a dataset where the value of feature i has been randomly rearranged while preserving the values of other features.

  This procedure is commonly iterated for every feature within the dataset, enabling the ranking of features according to their permutation importance values [108].

In our research, we apply the permutation method to datasets D1 and D2 using a basic random forest model to identify important features. As detailed in Section 2, each variant of the dataset comprises 111 features described in Tables 4.4 and A.1. Figures 4.16 and 4.17 exhibit bar charts illustrating the hierarchy of feature significance determined by the rise in model error for both D1 and D2, correspondingly. In this visualization, attributes with longer bars are considered more significant because their shuffling led to notably higher error values within the model. We regard all features producing a positive error value as significant while discarding those with an error value equal to or less than 0. As a result, the most notable features for each dataset, as ranked by the permutation importance method in our study, are presented in Tables 4.14 and 4.15, respectively, resulting in 32 top features for D1 and 30 top features for D2 (more details about the symbols of these features and their description are listed in tables 4.4 and A.1). Ultimately, we select the most relevant and influential features to proceed to the next phase of our methodology [108].

- **Train Test splitting**

  Employing the sci-kit-learn Python library alongside the "train test split" function, both datasets are partitioned into training and testing sets, with an 80% allocation for training and 20% for testing. After splitting the dataset D1, which initially contains 58,645 samples before data balancing, it results in 46,916 samples for training and 11,729 samples for testing. After employing the balancing method, the dataset grows to 58,400 samples, resulting in 46,720 samples for training and 11,680 samples for testing.

  Similarly, after splitting dataset D2, which contains 88,647 samples, it yields 70,917 samples for training and 17,730 samples for testing. After applying data balancing, the dataset expands to 113,210 samples, resulting in 90,568 samples for training and 22,642 samples for testing.

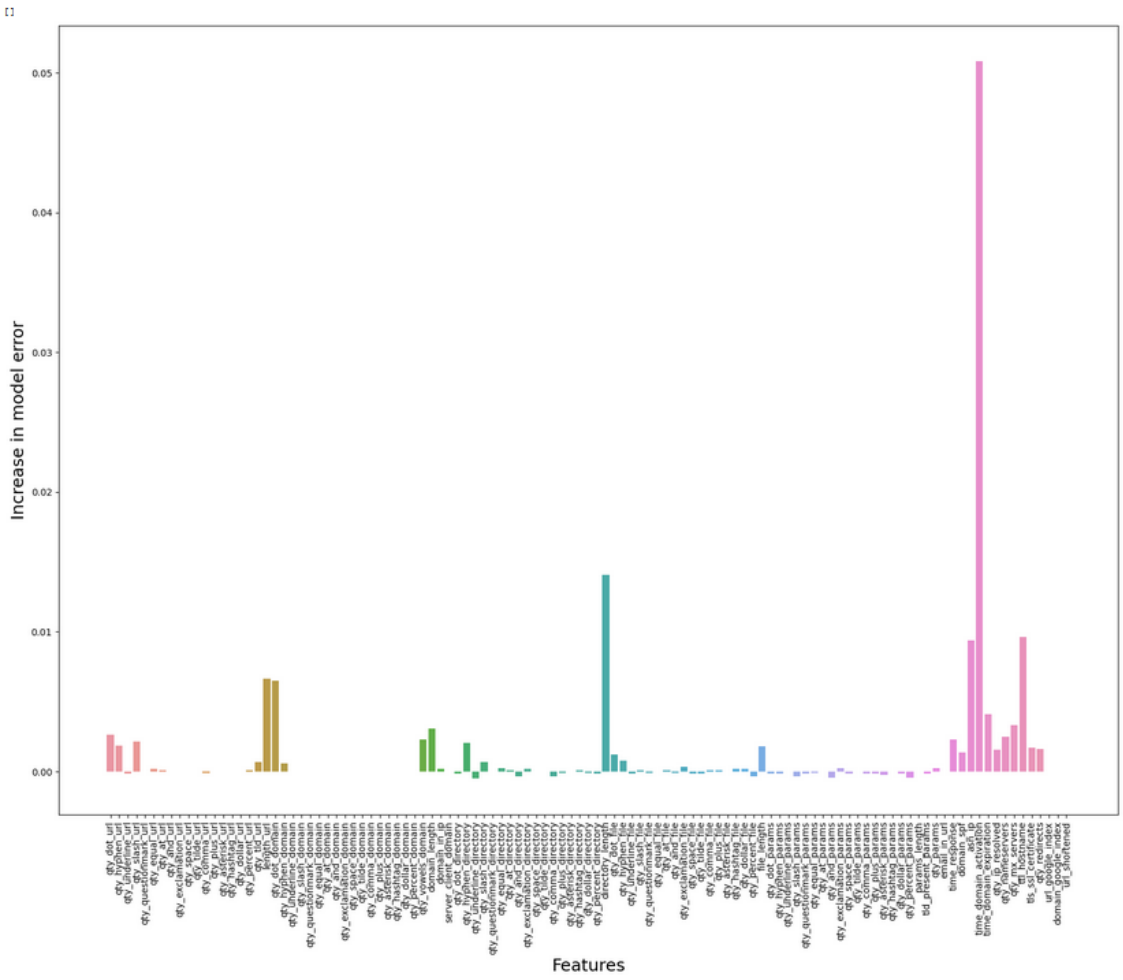


Figure 4.16: The feature importance ranking based on the increment in model error using the permutation importance method applied on the D1 dataset [108].

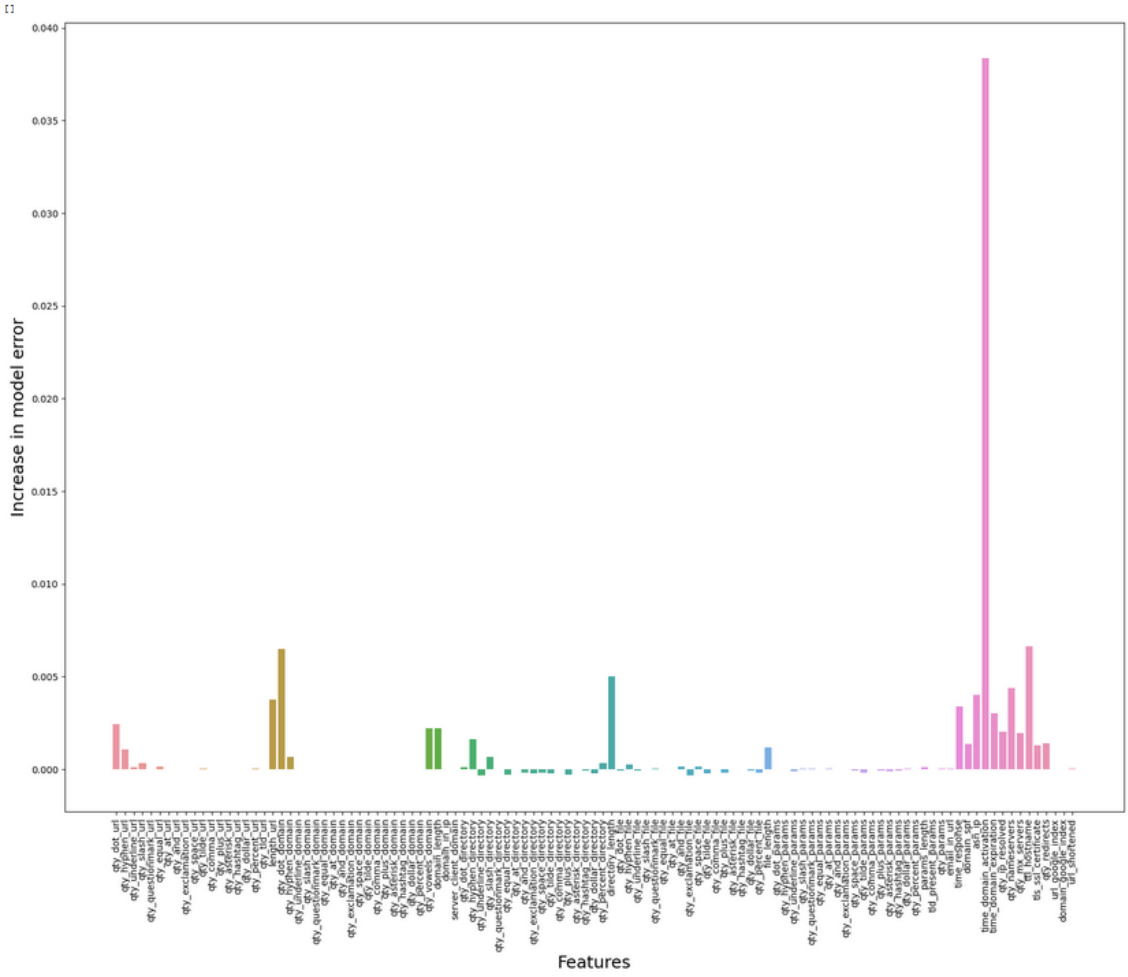


Figure 4.17: The feature importance ranking based on the increment in model error using the permutation importance method applied on the D2 dataset [108].

Table 4.14: The 32 most crucial features identified through the PIM in our study for dataset D1 [108].

Top selected features for D1
qnty-of-dot-in-url
qnty-of-hyphen-in-url
qnty-of-slash-in-url
qnty-of-percent-in-url
qnty-of-tld-in-url
length-of-url
qnty-of-dot-in-domain
qnty-of-hyphen-in-domain
qnty-of-vowels-in-domain
length-of-domain
if-domain-in-IP
qnty-of-hyphen-in-directory
qnty-of-slash-in-directory
qnty-of-equal-in-directory
qnty-of-exclamation-in-directory
length-of-directory
qnty-of-dot-in-file
qnty-of-hyphen-in-file
qnty-of-exclamation-in-file
length-of-file
qnty-of-exclamation-in-params
time-of-response
the-domainnspf
asn-IP
time-of-domainactivation
time-of-domainexpiration
qnty-of-IP-resolved
qnty-of-nameservers
qnty-of-mx-servers
ttl-hostnamee
tls-ssl-certificatee
qnty-of-redirects

Table 4.15: The 30 most crucial features identified through the PIM in our study for dataset D2 [108].

Top selected features for D2
qnty-of-dot-in-url
qnty-of-hyphen-in-url
qnty-of-under-line-in-url
qnty-of-slash-in-url
qnty-of-equal-in-url
length-of-url
qnty-of-dot$_i n - domain$
qnty-of-hyphen-in-domain
qnty-of-vowels-in-domain
length-of$_d omain$
qnty-of-dot-in-directory
qnty-of-hyphen-in-directory
qnty-of-slash-in-directory
qnty-of-percent-in-directory
length-of-directory
qnty-of-hyphen-in-file
qnty-of-and-in-file
qnty-of-space-in-file
length-of-file
time-of-response
the-domainspf
asn-IP
time-of-domainactivation
time-of-domainexpiration
qnty-of-ip-resolved
qnty-of-nameservers
qnty-of-mx-servers
ttl-hostnamee
tls-ssl-certificatee
qnty-of-redirects

4.4.1.2 The models used

This section presents the proposed classification models for the mechanism of phishing detection. CNNs and LSTMs are widely utilized in the domain of identifying deceptive websites, consistently demonstrating notable efficacy. LSTMs are particularly adept at capturing extended relationships in sequential data, whereas CNNs excel at identifying localized patterns within datasets [123].
Five distinct classifiers were utilized for training and testing: XGboost, CNN, LSTM, as well as Hybrid models, which include both CNN-LSTM and LSTM-CNN architectures. The choice of these classifiers stems from the literature review, which indicates that these two DL classifiers yielded the best results and are most suitable for phishing detection. Moreover, XGBoost is frequently applied in research addressing the challenge of identifying and categorizing attacks on imbalanced datasets. Its boosting and sequential tree-building features make XGBoost especially potent for handling imbalanced datasets. [101], [108].

1. **The XGBoost classifier**
   XGBoost is an ensemble learning approach rooted in boosting, a methodology that progressively refines the model to handle misclassified data with each iteration. Within XGBoost, decision trees are constructed sequentially and fine-tuned to minimize a designated loss function. Each subsequent tree within the ensemble aims to rectify the mistakes of its predecessors. XGBoost utilizes boosting, a technique where weak learners (basic decision trees) are amalgamated to create a robust learner (a sophisticated ensemble model). [101].
   XGBoost extends the principles of gradient boosting. The algorithm comprises two primary elements: a weak learner model, commonly a decision tree, and an additive model that amalgamates numerous weak learners. The ultimate model is a weighted aggregation of these weak learners, with each tree incorporated into the model in sequence [101], [108].

   The equation (eq 4.7) for the predicted output of the XGBoost model can be expressed as:

$$\hat{y}_i = \phi(x_i) = \sum_{k=1}^{K} f_k(x_i) \tag{4.7}$$

   where:

   $\hat{y}_i$ represents the predicted output for the ith sample,

   x_i denotes the feature vector for the$i$th sample,

   $\phi(x_i)$signifies the final predicted output, obtained by summing the predictions of all weak learners,

   $f_k(x_i)$signifies the prediction of the kth weak learner (tree) for the ith sample,

   K stands for the total number of weak learners in the ensemble.

Usually, each weak learner $f_k(x_i)$ is a decision tree, and the prediction is generated by navigating the tree from its root to a leaf node. The output value linked with that leaf node is then utilized as the prediction [101].

2. **The CNN Classifier**
   The model is constructed around a Convolutional Neural Network (CNN), a sophisticated, feed-forward artificial neural network. It comprises an input layer, followed by two convolutional layers, each accompanied by a batch normalization and max pooling layer. The output from the max pooling layer is then flattened and fed into two dense layers to generate an output with two categories (phishing or legitimate). Dropout layers are incorporated after each max pooling layer and following the first dense layer to address overfitting concerns. Figure 4.18 provides a detailed depiction of the CNN model's architecture.
   Optimizing the model's performance significantly relies on selecting an appropriate set of parameters and fine-tuning them. Hence, for each model employed in this study, parameters were chosen based on their effectiveness in improving model performance after a series of experiments. Table 4.16 outlines the parameters defined during the design of the CNN model [108].

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 110, 32)           96

 batch_normalization (Batch  (None, 110, 32)           128
 Normalization)

 max_pooling1d (MaxPooling1  (None, 55, 32)            0
 D)

 dropout (Dropout)           (None, 55, 32)            0

 conv1d_1 (Conv1D)           (None, 54, 64)            4160

 batch_normalization_1 (Bat  (None, 54, 64)            256
 chNormalization)

 max_pooling1d_1 (MaxPoolin  (None, 27, 64)            0
 g1D)

 dropout_1 (Dropout)         (None, 27, 64)            0

 flatten (Flatten)           (None, 1728)              0

 dense (Dense)               (None, 64)                110656

 dropout_2 (Dropout)         (None, 64)                0

 dense_1 (Dense)             (None, 1)                 65

=================================================================
Total params: 115361 (450.63 KB)
Trainable params: 115169 (449.88 KB)
Non-trainable params: 192 (768.00 Byte)
```

Figure 4.18: Summary of the CNN model architecture [108].

Table 4.16: The optimal parameters set used during the design of the CNN model [108].

Parameter	Value
layer of 1D CNN	Filters=32, Kernel size=2, Activation function=relu
layer of BatchNormalization	1
Size of 1D MaxPooling	2
Dropout layer around	0.5
1D CNN layer	Filters=64, Kernel size=2, Activation function=relu
layer of BatchNormalization	1
Size of 1D MaxPooling	2
Dropout layer around	0.5
The Flatten layer	1
The Dense layer	64 neurons, Activation function=relu
Layer of Dropout	0.5
The Dense layer	1 neuron, Activation function= sigmoid
Batch size	32
The used Optimizer	Adam
The Learning rate	0.001
The Loss function	Binary_crossentropy
Number of Epochs	50

3. **The LSTM Classifier**

   The LSTM archcitecture adopted in this study is depicted in Figure 4.19. It comprises two Long Short-Term Memory (LSTM) layers followed by two dense layers. The initial dense layer encompasses 64 units, while the subsequent dense layer consists of one unit using the sigmoid activation function. The model utilizes the Adam optimizer and binary cross-entropy as the loss function.

   To conform to the LSTM model's input format, the input data is reshaped accordingly. Table 4.17 provides the optimal parameters set established during the implementation of the LSTM model [108].

```
Model: "sequential_20"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_21 (LSTM)              (None, 1, 32)             18432

 dropout_7 (Dropout)         (None, 1, 32)             0

 lstm_22 (LSTM)              (None, 32)                8320

 dropout_8 (Dropout)         (None, 32)                0

 dense_24 (Dense)            (None, 64)                2112

 dense_25 (Dense)            (None, 1)                 65

=================================================================
Total params: 28929 (113.00 KB)
Trainable params: 28929 (113.00 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 4.19: Summary of the LSTM model architecture [108].

Table 4.17: The optimal parameters set used during the design of the LSTM model [108].

Parameter	Value
Filters of the LSTM layer	32
Dropout layer around	0.5
Filters of the LSTM layer	32
Dropout layer around	0.5
The Dense layer	64 neurons
The Dense layer	Neuron=1, Activation function=sigmoid
The Batch size	32
The used Optimizer	Adam
The Learning rate	0.001
The Loss function	Binary_crossentropy
Number of Epochs	50

4. **Hybrid models (CNN-LSTM & LSTM-CNN)**

   In our research, we opted to investigate two hybrid models: LSTM followed by CNN (CNN-LSTM) and CNN followed by LSTM (LSTM-CNN). The rationale behind this decision is that hybrid models address the limitations of each individual model and leverage the

strengths of both to enhance detection performance. We tested two variations of the hybrid model to determine the superior one. In the CNN-LSTM model, the output of the CNN serves as the input to the LSTM. This configuration allows the LSTM to learn features from the input data identified by the CNN. While the CNN focuses on capturing local patterns, the LSTM can capture temporal dependencies and make final predictions. Figure 4.20 depicts the proposed architecture of the CNN-LSTM model. It comprises three 1D convolutional layers with ReLU activation function, three batch normalization layers, two max-pooling layers, two LSTM layers, and two dense layers. Dropouts are incorporated after each max-pooling layer and following the first dense layer to address overfitting.
Figure 4.21 provides the architecture of the LSTM-CNN, which shares the same structure as the CNN-LSTM but in reverse order. It begins with a single LSTM layer and includes a flatten layer before the first dense layer. Tables 4.18 and 4.19 display the optimal parameter configurations determined during the implementation of each hybrid model [108].

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 110, 32)           96

 batch_normalization (Batch  (None, 110, 32)           128
 Normalization)

 dropout (Dropout)           (None, 110, 32)           0

 conv1d_1 (Conv1D)           (None, 109, 64)           4160

 batch_normalization_1 (Bat  (None, 109, 64)           256
 chNormalization)

 max_pooling1d (MaxPooling1  (None, 54, 64)            0
 D)

 dropout_1 (Dropout)         (None, 54, 64)            0

 conv1d_2 (Conv1D)           (None, 53, 64)            8256

 batch_normalization_2 (Bat  (None, 53, 64)            256
 chNormalization)

 max_pooling1d_1 (MaxPoolin  (None, 26, 64)            0
 g1D)

 dropout_2 (Dropout)         (None, 26, 64)            0

 lstm (LSTM)                 (None, 26, 64)            33024

 lstm_1 (LSTM)               (None, 64)                33024

 dense (Dense)               (None, 64)                4160

 dropout_3 (Dropout)         (None, 64)                0

 dense_1 (Dense)             (None, 1)                 65

=================================================================
Total params: 83425 (325.88 KB)
Trainable params: 83105 (324.63 KB)
Non-trainable params: 320 (1.25 KB)
_________________________________________________________________
```

Figure 4.20: Summary of the CNN-LSTM model architecture [108].

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_2 (LSTM)               (None, 30, 64)            16896

 conv1d_3 (Conv1D)           (None, 29, 32)            4128

 batch_normalization_3 (Bat  (None, 29, 32)            128
 chNormalization)

 dropout_4 (Dropout)         (None, 29, 32)            0

 conv1d_4 (Conv1D)           (None, 28, 64)            4160

 batch_normalization_4 (Bat  (None, 28, 64)            256
 chNormalization)

 max_pooling1d_2 (MaxPoolin  (None, 14, 64)            0
 g1D)

 dropout_5 (Dropout)         (None, 14, 64)            0

 conv1d_5 (Conv1D)           (None, 13, 64)            8256

 batch_normalization_5 (Bat  (None, 13, 64)            256
 chNormalization)

 max_pooling1d_3 (MaxPoolin  (None, 6, 64)             0
 g1D)

 dropout_6 (Dropout)         (None, 6, 64)             0

 flatten (Flatten)           (None, 384)               0

 dense_2 (Dense)             (None, 64)                24640

 dropout_7 (Dropout)         (None, 64)                0

 dropout_8 (Dropout)         (None, 64)                0

 dense_3 (Dense)             (None, 1)                 65

=================================================================
Total params: 58785 (229.63 KB)
Trainable params: 58465 (228.38 KB)
Non-trainable params: 320 (1.25 KB)
_________________________________________________________________
```

Figure 4.21: Summary of the LSTM-CNN model architecture [108].

Table 4.18: The optimal parameters set used during the design of the CNN-LSTM model [108].

Parameter	Value
Layer of the 1D Conv	Filters=32, Kernel size=2, Activation function=relu
Layer of BatchNormalization	1
Dropout layer around	0.25
Layer of 1D Conv	Filters=64, Kernel size=2, Activation function=relu
Layer of BatchNormalization	1
Kernel size of 1D MaxPooling	2
Dropout	0.5
Layer of 1D Conv	Filters=64, Kernel size=2, Activation function=relu
Layer of BatchNormalization	1
Kernel size of 1D MaxPooling	2
Dropout	0.5
Layer of LSTM	64
Layer of LSTM	64
The Flatten layer	1
The Dense layer	64 neurons, Activation function=relu
The Dropout layer	0.5
The Dense layer	1 neuron, Activation function= sigmoid
The Batch size	64
The used Optimizer	Adam
The Learning rate	0.001
The Loss function	Binary_crossentropy
Number of Epochs number	50

Table 4.19: The optimal parameters set used during the design of the LSTM-CNN model [108].

Parameter	Value
Layer of LSTM	64
Layer of 1D Conv	Filters=32, Kernel size=2, Activation function=relu
Layer of BatchNormalization	1
Dropout layer around	0.25
Layer of 1D Conv	Filters=64, Kernel size=2, Activation function=relu
Layer of BatchNormalization	1
Kernel size of 1D MaxPooling	2
Dropout	0.5
Layer of 1D Conv	Filters=64, Kernel size=2, Activation function=relu
Layer of BatchNormalization	1
kernel size of 1D MaxPooling	2
Dropout	0.5
The Flatten layer	1
The Dense layer	64 neurons, Activation function=relu
The Dropout layer	0.5
The Dense layer	1 neuron, Activation function= sigmoid
The Batch size	64
The used Optimizer	Adam
The Learning rate	0.001
The Loss function	Binary_crossentropy
Number of Epochs	50

4.4.2 Implementation and Evaluation

This section outlines the implementation and evaluation of the phishing detection mechanism proposed in this study.

The experimentation took place in the same environments as mentioned in section 4.3.2.1, except this time we utilized Google Colab with Python programming language due to resource constraints. For the implementation of the web application, we employed Flask, a web framework, and Ngrok, a widely used gateway providing secure connectivity for applications and services in various environments. In our experiment, we assess the effectiveness of the proposed mechanism by employing it on two variations of the Mendeley dataset (referred to as D1 and D2)(sect 2. Each classifier is trained with the training dataset and subsequently evaluated using both the testing dataset, with and without feature selection, and data balancing. The outcomes, based on the performance metrics detailed in Section 4.2.1.4, are presented for each scenario across the two datasets [108].

4.4.2.1 Results and Analysis

Table 4.20 illustrates the outcomes of all the classifiers introduced in this research, both with and without feature selection and data balancing, on dataset D1. The findings presented in the table demonstrate that the hybrid model (CNN-LSTM) achieved a remarkable accuracy of 95.14%, while the XGBoost classifier attained a notable accuracy of 96.10%. This level of performance was achieved through the implementation of data balancing and feature selection, surpassing not only other classifiers but also their performance when feature selection and data balancing were not applied.

The outcomes of dataset D2 are outlined in Table 4.21, showcasing the performances of all classifiers introduced in this study with and without feature selection and data balancing. The table highlights that all classifiers achieved significant accuracies exceeding 95.99% when employing feature selection and data balancing. Particularly, both LSTM and the hybrid model (CNN-LSTM) achieved substantial accuracies of 96.88%, while the XGBoost classifier attained the highest accuracy value of 97.82%. This performance surpasses their respective accuracies on imbalanced data and without feature selection, as well as outperforms other classifiers.

The accuracy and loss graphs depicting the performance of the four deep learning classifiers on datasets D1 and D2, both pre-and post-feature selection and data balancing, are depicted in Figures 4.22 and 4.23, respectively. Observing these graphs, we note a convergence between the training and validation curves of accuracy and loss as epochs progress. Additionally, there is no significant evidence of overfitting. These observations underscore the effective functioning of the proposed classifiers in detecting phishing websites [108].

Table 4.20: The results acquired from each classifier both before and after data balancing and feature selection on dataset D1 [108].

	Measures (%)	**XGBoost**	**CNN**	**LSTM**	**CNN-LSTM**	**LSTM-CNN**
Pre-data balancing and features selection (D1(111))	**Accuracy**	95.30	93.36	93.90	94.27	94.26
	Precision	95.31	91.73	93.07	93.17	92.68
	Recall	94.74	94.48	94.12	94.84	95.42
	F1-Score	95.02	93.09	93.60	94.00	94.02
Post data balancing and features selection (D1(32))	**Accuracy**	**96.10**	94.12	94.66	95.14	94.77
	Precision	96.42	94.42	94.18	95.00	94.62
	Recall	95.93	94.06	95.45	95.51	95.18
	F1-Score	96.17	94.23	94.80	95.25	94.91

Table 4.21: The results acquired from each classifier both before and after data balancing and feature selection on dataset D2 [108].

	Measures (%)	XGBoost	CNN	LSTM	CNN-LSTM	LSTM-CNN
Pre-data balancing and features selection (D2(111))	**Accuracy**	97.05	95.71	96.10	96.14	96.43
	Precision	97.76	96.06	96.46	96.90	96.99
	Recall	97.71	97.41	97.59	97.18	97.54
	F1-Score	97.73	96.73	97.02	97.04	97.27
Post data balancing and features selection (D2(30))	**Accuracy**	**97.82**	95.99	96.88	96.88	96.82
	Precision	98.03	96.92	96.85	97.23	97.27
	Recall	97.62	95.04	96.94	96.54	96.37
	F1-Score	97.82	95.97	96.89	96.88	96.82

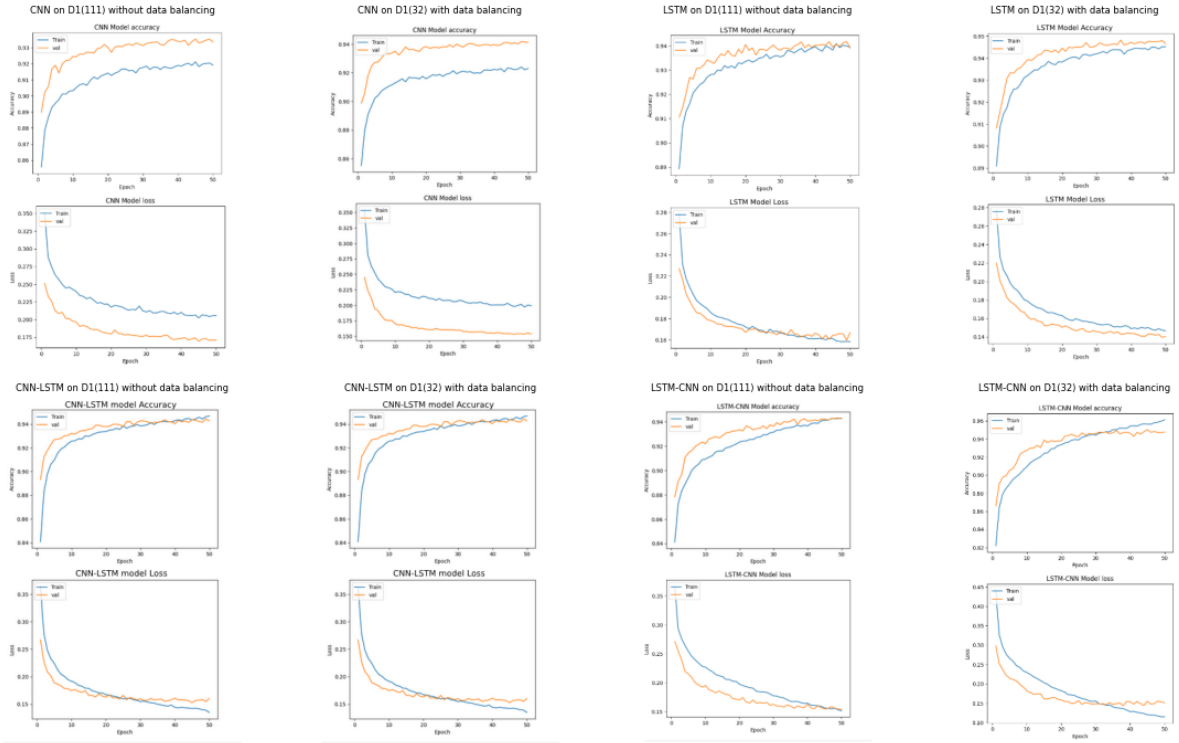


Figure 4.22: Graphs depicting accuracy and loss of the four DL classifiers on dataset D1, pre and post-feature selection and data balancing [108].

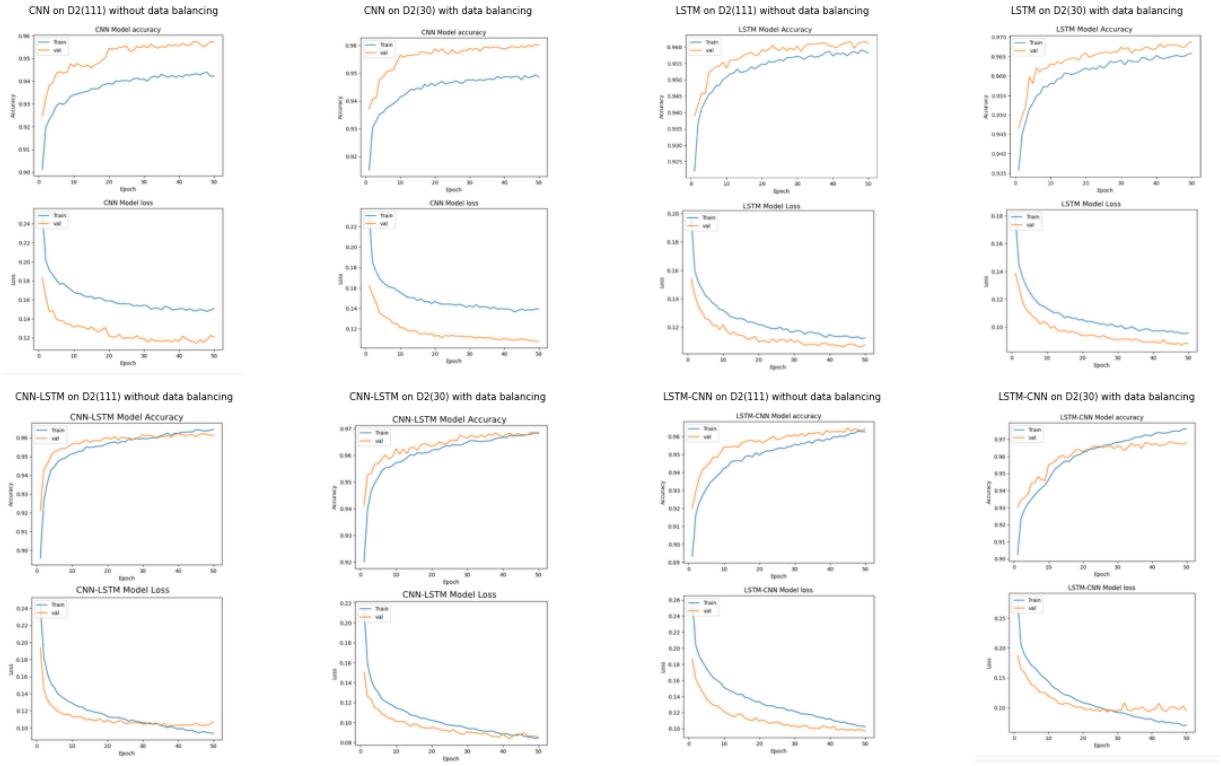


Figure 4.23: Graphs depicting accuracy and loss of the four DL classifiers on dataset D2, pre and post-feature selection and data balancing [108].

4.4.2.2 Implementation of the proposed mechanism as a Web application

The phishing detection mechanism proposed has been implemented as a web application to amplify its accessibility for internet users. The advantage lies in the fact that the underlying model operating behind the web interface is built upon deep learning that can detect novel attacks, often referred to as zero-day attacks.

Since the proposed approach was implemented on Google Colab using Python programming, we utilized the Flask and Ngrok Python modules to create the web application. Flask is a lightweight and versatile Python web framework that equips developers with the necessary tools and libraries to construct web applications swiftly and effectively. It serves as a Python module facilitating the development of web applications with ease [102]. To enable access to the Flask application from a web browser, we leveraged a functionality known as "Ngrok," (Network grok) which establishes a secure tunnel to the Colab runtime. Ngrok furnishes a publicly accessible URL that redirects traffic to the local server.

The code depicted in Figure 4.24 presents a Ngrok tunnel, exposing a Flask application running

locally to the internet.

- **Setting Ngrok AuthToken:**
  ‘ngrok.set_auth_token(’2bkGUaymJj6jcpLIhEKqo0LJQvY_4PjoY5VmdUq7eYBQx9HWV’)‘ configures the authentication token for Ngrok, imperative for utilizing its features.
  ‘public_url = ngrok.connect(5000)‘ establishes a Ngrok tunnel on local port 5000, where the Flask application resides. Ngrok allocates a public URL to this tunnel, granting external users access to the local Flask application. The ‘public_url‘ variable retains the URL of the Ngrok tunnel, enabling access to our Flask application from the internet, as demonstrated in Figure 4.25, which illustrates the source code implementing the Flask application. This code encompasses various steps, including the utilization of index.html, an HTML file stored in Google Drive, containing the interface code of the web application crafted with HTML, Javascript, and CSS. Additionally, the trained model saved in Google Drive is utilized for prediction. As depicted, the URL undergoes preprocessing steps before being forwarded to the model for prediction. The ‘preprocess_input‘ function extracts the same features from the URL as those the model was trained with. (In this case, we tested the hybrid model with D2(30)).
  Figures 4.26 and 4.27 showcase the execution of our web application in two scenarios: legitimate URL detection and phishing URL detection, respectively.

```
from pyngrok import ngrok

# Set your ngrok authtoken
ngrok.set_auth_token('2bkGUaymJj6jcpLIhEKqo0LJQvY_4PjoY5VmdUq7eYBQx9HWV')



# Set up ngrok tunnel
public_url = ngrok.connect(5000)
public_url
```

Figure 4.24: Establishment of the Ngrok tunnel.

```python
from flask import Flask, render_template, request, jsonify
import pickle
import numpy as np
app = Flask(__name__)
run_with_ngrok(app)
# Load the trained model from drive (in this case we tested with the Hybrid model)
model_filename = '/content/drive/MyDrive/Project/Hybridmodel.pkl'
with open(model_filename, 'rb') as file:
    model = pickle.load(file)
#The index.html file contain the interface code of the web app using html, jabascript and CSS
# Home route
@app.route('/')
def home():
    return render_template('index.html')
# Prediction route
@app.route('/predict', methods=['POST'])
def predict():
    try:
        if request.method == 'POST':
            url = request.form['url']
            # preprocessing steps before introduce the URL to the model
            processed_input = preprocess_input(url)   #this function extract features from the URL in this case we used 30 features of D2
            processed_input_values = list(processed_input.values())
            processed_input_values_reshaped = np.array(processed_input_values).reshape(-1, 1)
            processed_input_scaled = scaler.fit_transform(processed_input_values_reshaped)
            processed_input_final = np.array(processed_input_scaled).reshape(1, 30, 1)
            # Make a prediction
            prediction = model.predict(processed_input_final)
            # Display the result
            result = 'It is a Phishing Website!' if prediction < 0.5 else 'It is a legitimate Website'

            # Return a JSON response
            return jsonify({'prediction_result': result, 'url': url})

    except Exception as e:
        # Log the full traceback for debugging
        import traceback
        traceback.print_exc()
        # Return an error response
        return jsonify({'error': str(e)})
if __name__ == '__main__':
    app.run()
```

Figure 4.25: The Python source code of the web application.

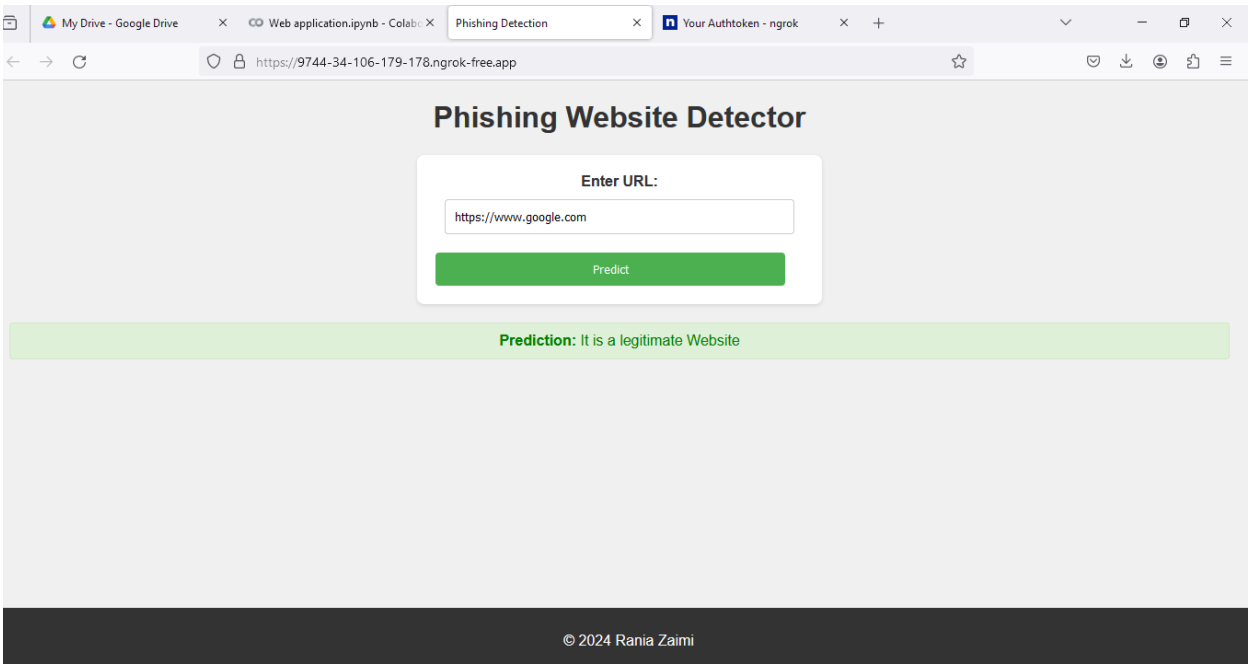


Figure 4.26: Running the web application in case of legitimate detection.

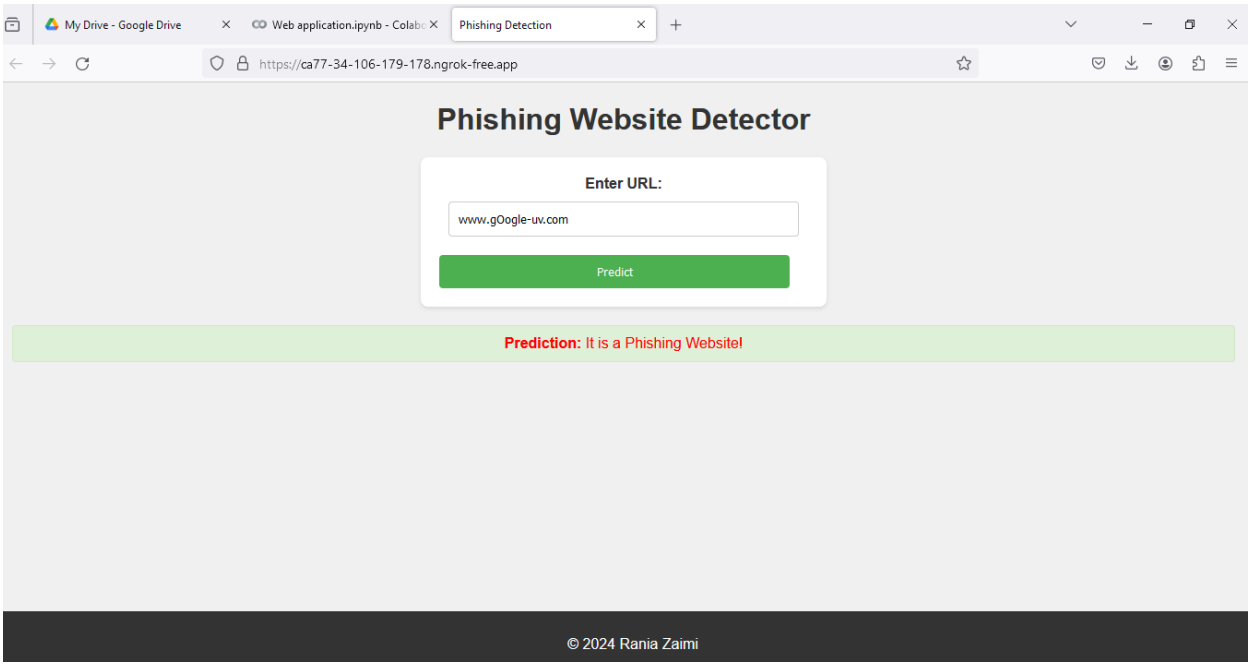


Figure 4.27: Running the web application in case of phishing detection.

4.4.2.3 Discussion and comparison with literature

In this section, we discuss the obtained results and conduct a comparative analysis between our proposed model and previous research efforts.
Our experiments yielded noteworthy performance across both scenarios with datasets D1 and D2. Upon analyzing the results of the aforementioned classifiers, it is evident that all the performance metrics values saw improvement across the five classifiers when the data balancing and features selection methods were employed. Furthermore, it has been observed the LSTM and the hybrid model (CNN-LSTM) achieved considerable accuracy values in our study, reaching nearly 97% when using the D2 dataset. Furthermore, XGBoost achieves the highest accuracy rate of 97.82%, precision rate of 98.03%, recall rate of 97.71%, and F1-score of 97.82%, indicating its suitability for the phishing dataset used. The incorporation of permutation importance feature selection and SMOTE-Tomek Link data balancing techniques notably enhances the efficacy of the phishing detection approach, making it more resilient [108].

Our model was tested on two datasets, and we specifically compared our approach with studies that utilized the same dataset. Notably, due to a lack of prior research applied to the D1 dataset, we are unable to provide comparative results with existing studies for this dataset. Regarding Dataset D2, we refer to existing work cited as [47], [1], and [24]. Table 4.22 and Figure 4.28 provide a comparison of the results obtained from our experiments, incorporating feature selection and data balancing, in contrast to findings from existing literature [108].

From the outcomes, it is evident that our introduced mechanism employing XGBoost, LSTM, and the hybrid model (CNN-LSTM) outperformed all the findings of existing works with significant accuracies. In addition, The CNN and LSTM-CNN models outperformed studies in [47] and [1] in terms of accuracy. These findings collectively validate the effectiveness of the proposed mechanism in accurately detecting phishing websites [108].

Table 4.22: Comparison of performance between the proposed classifiers and prior studies utilizing the D2 phishing dataset [108].

	XGBoost	**CNN**	**LSTM**	**CNN-LSTM**	**LSTM-CNN**	**[47]**	**[1]**	**[24]**
F S method	PIM	PIM	PIM	PIM	PIM	/	BSSA	DEFSTH
Accuracy (%)	**97.82**	95.99	96.88	96.88	96.82	94.03	95.07	96.82
Precision (%)	98.03	96.92	96.85	97.23	97.27	/	/	/
Recall (%)	97.62	95.04	96.94	96.54	96.37	/	/	/
F1-Score (%)	97.82	95.97	96.89	96.88	96.82	/	/	95.38

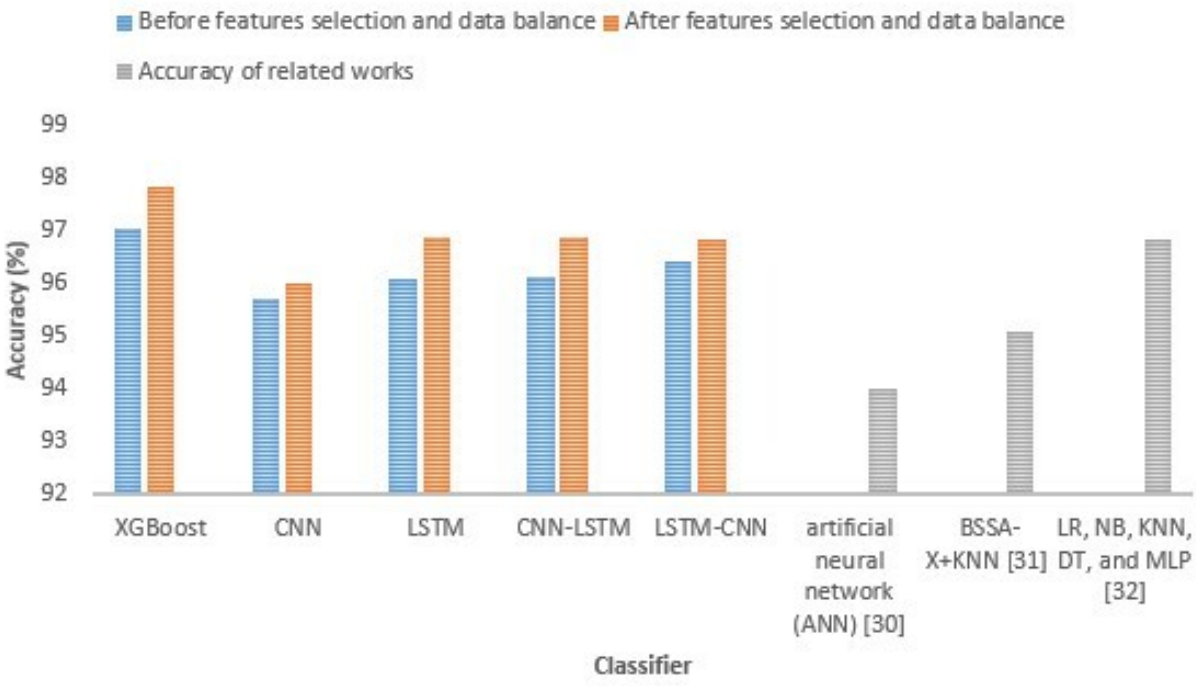


Figure 4.28: Comparison graph depicting the accuracy of the proposed classifiers in our study compared to existing literature utilizing the D2 dataset [108].

4.5 Conclusion

In this chapter, we proposed two intelligent systems for highly efficient and high-performance phishing website detection. These mechanisms are based on deep learning and preprocessing techniques that help enhance detection performance and meet the following requirements: they are based on emerging technologies that enable the detection of zero-day phishing attacks. For the first contribution, it is very effective in phishing website detection and helps reduce training time. For the second contribution, the use of different classifiers and both preprocessing techniques (SMOTE-Tomek link) and PIM made the phishing detection system more robust.
Experiments on two databases from the Mendeley "Web page phishing detection dataset" [60] and "Phishing Websites Dataset" [133] showed that our models provide the highest detection rate compared to related works that used the same datasets and other well-known works. Furthermore, the implementation of our approach as a web application helps facilitate usage for web users to protect them in real time from phishing attacks. This web application can even be used with the first approach. Finally, these two contributions based on deep learning and machine learning can be used individually or combined as a multi-filter detection system to enhance detection and provide additional cybersecurity measures.

Chapter. 5

Third Contribution: An enhanced deep learning approach to detect Malicious URLs based on DistilBERT features extraction

5.1 Introduction

Recently, numerous methodologies have emerged for identifying cyber phishing URLs by utilizing both machine learning and deep learning. There has been a notable enhancement in the detection performance. Furthermore, transfer learning, emerging as a novel deep learning approach, has garnered significant interest among researchers, showcasing promising outcomes across various natural language processing tasks. The primary objective of this study is to combine transformer learning and deep learning methodologies for the detection of malicious websites used by phishers. Phishing involves deceitful tactics where individuals and organizations are tricked into visiting malicious URLs and divulging sensitive information. This approach aims to overcome limitations observed in machine learning and conventional phishing detection techniques, particularly in addressing challenges associated with natural language processing. By combining these methods, the goal is to boost classification performance in detecting malicious URLs.
Different language models are employed for tasks involving text classification. In this chapter, we introduce an enhanced approach for detecting phishing URLs utilizing the DistilBERT method for feature extraction and Deep Learning for classification. DistilBERT is utilized to extract features from URLs and capture relevant textual information. Subsequently, a hybrid deep learning model (CNN-LSTM) is utilized to classify the dataset into malicious and benign URLs. Using DistilBERT aids in extracting semantic and syntactic features from the URL, thereby preparing the input text to be introduced to the DL model and potentially enhancing overall performance.

To evaluate the proposed approach, we utilized two public datasets, namely the "Phishing Site URLs dataset" and the "Malicious URLs dataset," both of which are considerably large and sourced from Kaggle. Across these two datasets, we conducted two scenarios: first, introducing

the word embeddings alone to the classification model, and secondly, incorporating 21 URL features alongside the word embeddings, subsequently introducing them to the classification model to enhance classification performance. The experimental results demonstrated that the proposed method achieved outstanding performance, surpassing relevant approaches in the literature employing similar datasets. This chapter presents an enhanced approach that offers several key contributions:

- Development of an intelligent deep learning method leveraging DistilBERT for feature extraction to identify relevant features for detecting malicious URLs.

- Improvement in the accuracy of malicious website detection by combining two emerging technologies: deep learning and transfer learning, with the use of DistilBERT being a novel contribution in this field.

- The proposed approach was subjected to comprehensive comparison, analysis, and evaluation, which highlighted its performance and effectiveness. Additionally, it was benchmarked against alternative methods and findings from related literature using similar datasets, demonstrating superior performance over other methods in the literature.

This chapter was the subject of our paper titled "An enhanced mechanism for malicious URL detection using deep learning and DistilBERT-based feature extraction". J Supercomput 81, 438 (2025) [109].
Section 5.2 describes the proposed approach, section 5.3 provides its implementation and evaluation of performances, and section 5.4 concludes this chapter.

5.2 Description of the overall proposed approach

This study proposes an enhanced anti-phishing approach that utilizes Natural Language Processing via the DistilBERT transformer to detect malicious URLs. DistilBERT is a distilled version of the BERT model and is employed alongside advanced deep-learning models to improve the classification process. Figure 5.1 depicts a detailed block diagram of the proposed methodology, consisting of four stages presented in the following subsections. The figure presents a detailed block diagram outlining the key components and flow of the proposed anti-phishing methodology based on DistilBERT transfer learning. The diagram visually depicts the four main stages of the proposed approach, providing a clear overview of the process from input to output [109]. These steps consist of:

1. The dataset selection stage, explains the datasets used and the reasons for choosing them.

2. The data preprocessing stage, explains the several steps applied in this step.

3. The feature extraction stage, explains how DistilBERT is used for extracting features.

4. The classification stage, in which we describe the hybrid deep learning approach used.

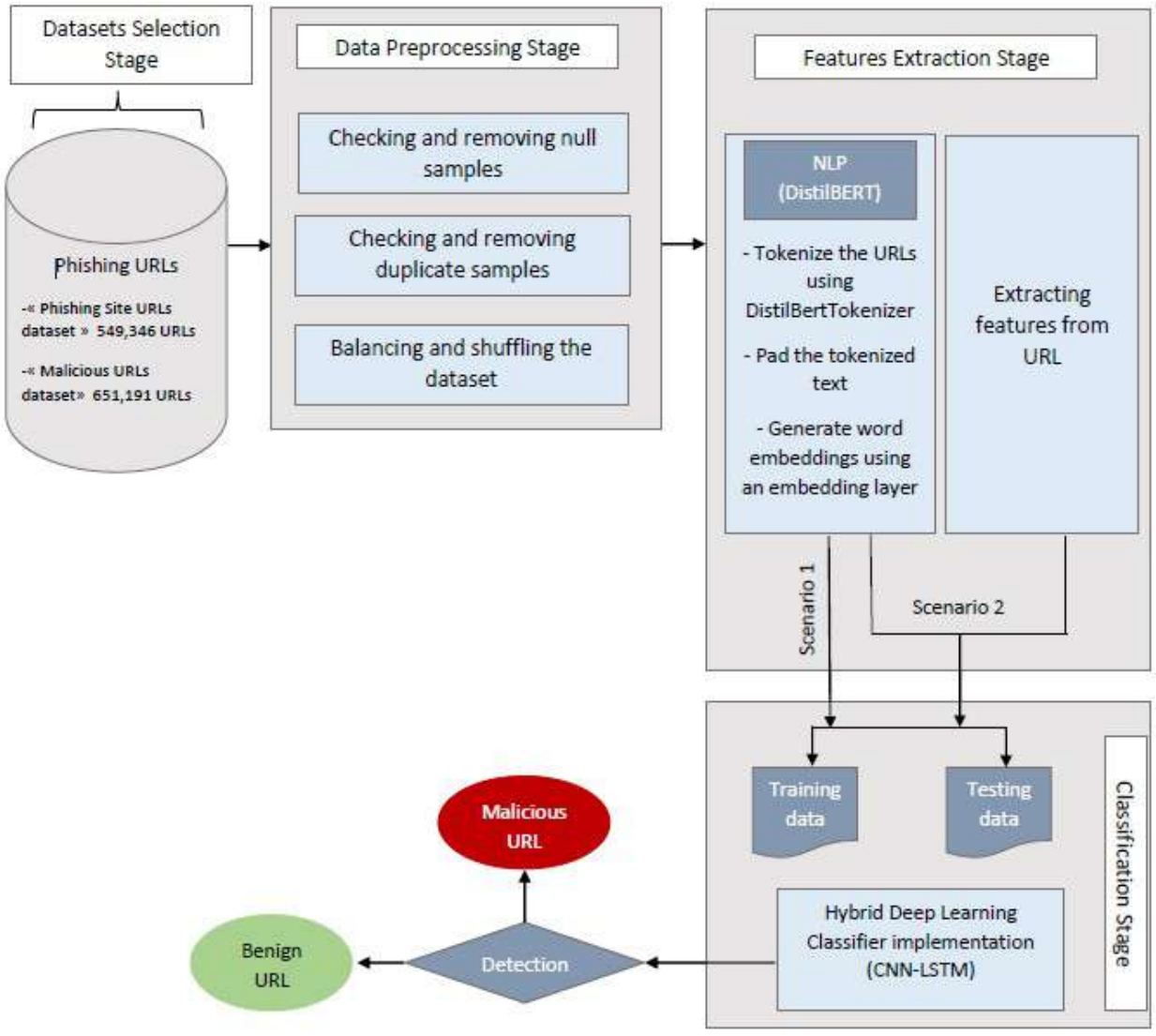


Figure 5.1: Block diagram of the third proposed approach [109].

5.2.1 Datasets selection Stage

At this stage, we selected the datasets to implement the proposed approach. We utilized two public datasets from Kaggle: the "Phishing Site URLs dataset" [125] and the "Malicious URLs dataset" [118]. We chose these datasets because they exclusively contain URLs and labels without any features, aligning well with the requirements for implementing NLP methods. Additionally, these datasets are notably large and publicly available, facilitating comparisons with findings in existing literature. Table 5.1 displays the size and distribution of these datasets, and further details about them are provided in subsection 4.2.1.1, items 3 and 4. As illustrated in the table, for the data 2, URLs categorized under 'defacement,' 'phishing,' and 'malware' have been consolidated and relabeled as 'malicious' URLs.

Table 5.1: Size and Distribution of the used datasets.

Data1		**Data2**		
Dataset name	"Phishing Site URLs Dataset" [125]	**Dataset name**		« Malicious URLs dataset» [118]
Overall Dataset size	549,346	**Overall Dataset size**		651,191
Good URLs size	392,924	**Benign URLs size**		428,103
Bad URLs size	156,422	**Malware**	32,520	**Malicious URLs** (223,088)
		Phishing	94,111	
		Defacement	96,457	

5.2.2 Data Preprocessing Stage

The efficacy of data preprocessing significantly impacts the outcome of any data analytics project. Upon reviewing the datasets, the 'type' column values were transformed into binary values, with 'bad/malicious' labeled as 1 and 'benign/good' as 0. The preprocessing steps adopted in this study are outlined as follows:

- Examination for null column values and elimination of corresponding rows. No null records were identified in the two datasets, maintaining the sample counts at 549,346 for data 1 and 651,191 for data 2.

- Verify for duplicate rows and eliminate them. Initially, as shown in Table 5.1, there were 549,346 samples in Data 1 before removing the duplicated URLs. After the removal of duplicate rows, the sample count was reduced to 507,195, indicating 42,151 duplicated rows. Similarly, for data 2, as illustrated in Table 5.1, the initial sample count was 651,191 before removing duplicates. Following the removal process, the sample count decreased to 641,119, indicating 10,072 duplicated rows.

- Ensure balance and shuffle the dataset. Table 5.1 illustrates a significant class imbalance in both datasets. Due to the nature of the dataset (text URL) and the techniques employed in our study (NLP), implementing balancing methods such as the Smote-Tomek link (like in our previous approach 4.4) was not feasible at this stage. Consequently, we opted for oversampling the minority class to achieve balance. This decision aimed to prevent data loss associated with undersampling while ensuring a sufficiently large dataset to support our proposed approach. However, to ensure the credibility of the results, balancing is applied only to the training set.

  For Data 1, the post-cleaning sample size was 507,195, with an initial distribution of 77% benign and 23% malicious. After splitting the data into training and testing sets with a ratio of 80:20, we applied data balancing to the training set using oversampling, achieving an equal distribution of 314,204 samples for both classes in the training dataset (train_df).

  For Data 2, after cleaning, the sample size was 641,119, with an initial distribution of 67% benign and 33% malicious. After splitting the data into training and testing sets with a ratio of 80:20, we applied data balancing to the training set using oversampling, achieving an equal distribution of 342513 samples for both classes in the training dataset (train_df). Both datasets were shuffled after balancing to prepare for the subsequent stages [109].

5.2.3 Features extraction Stage

5.2.3.1 Methods Used: Natural Language Processing

NLP encompasses the automated handling and analysis of human language, encompassing spoken, written, and electronic forms such as email or URL text. The intricacies of natural languages pose significant challenges due to their rule-defying nature and nuanced expressions, which even native speakers may struggle to comprehend accurately. Various approaches can be taken to tackle NLP, including linguistic analyses that delve into the syntax and semantics of language structures [121]. However, this thesis concentrates on employing statistical methods within NLP, particularly focusing on text data classification. In recent years, transfer learning has demonstrated remarkable efficacy in various natural language processing (NLP) tasks, including sentiment analysis, named entity recognition, and machine translation [62]. Popular language models such as ELMo, word2Vec, and GloVe have been extensively utilized to extract meaningful features for downstream tasks and subsequently classify them [89]. However, these conventional models often struggle with polysemic words and fail to capture intricate contextual relationships between words [89]. To address these limitations, BERT was introduced by Devlin et al. in 2018, and it comprises two variants: BERT (base) with a total parameter count of 110 million and BERT (large) with a total parameter count of 340 million. [43]. Unlike earlier pre-training models, BERT excels in capturing bidirectional context, leveraging two key strategies: the Masked Language Model (MLM) and Next Sentence Prediction (NSP). This innovative approach enables BERT to effectively understand and represent the contextual nuances of natural language text, leading to significant advancements in various NLP applications [73]. In recent years, BERT and its vari-

ants have become popular in natural language processing tasks. Some of the notable BERT-based models include BERT (Devlin et al., 2018) [43], Robustly optimized BERT approach (RoBERTa) (Liu et al., 2019) [85], A Lite BERT (ALBERT) (Lan et al., 2020) [80], Efficiently Learning an Encoder that Classifies Token Replacements Accurately (ELECTRA) (Clark et al., 2020) [36], and DistilBERT (Sanh et al., 2019) [116]. Each of these models offers unique improvements and optimizations over the original BERT architecture, making them suitable for various NLP applications.

In this thesis, we have opted to employ distilBERT for malicious website detection. During the feature extraction phase, we chose this model due to its compactness, speed, cost-effectiveness, and lightweight nature. This enables us to retain the knowledge from the BERT base model while utilizing a faster model suitable for resource-constrained computational environments. Given that we are working with significantly large datasets, these computational constraints are essential to consider [109]. Additionally, prior research has demonstrated promising results of this transformer model in the domain of cybersecurity [119].

5.2.3.2 DistilBERT

The concept of the DistilBERT model was introduced in the research conducted by Sanh et al. in 2019 [116]. DistilBERT, characterized as a compact, rapid, cost-effective, and lightweight Transformer model, is the result of distilling knowledge from the BERT base model. It boasts a reduction of 40% in parameters compared to the google-bert/bert-base-uncased variant, leading to a significantly smaller size. Furthermore, it exhibits a 60% increase in speed while retaining over 97% of BERT's language comprehension capabilities. DistilBERT emerges as the optimal solution for scenarios necessitating efficient operation within resource-constrained computational training or inference environments, particularly those on the edge of computational limitations [116].

DistilBERT represents a refined iteration of the BERT model, tailored for enhanced efficiency and reduced size. It embraces a technique known as knowledge distillation to achieve this optimization. This process involves training a smaller model by leveraging insights gleaned from the predictions and representations of a larger pre-trained model. Through knowledge distillation, the smaller model can effectively inherit the expertise and capabilities of its larger counterpart. This strategy is particularly beneficial for developing resource-efficient large language models. It entails fine-tuning a large BERT model on a specific task, identifying a sub-network, training a compact model, and then imparting knowledge from the larger model to the smaller one. The resulting DistilBERT model performs comparably to its larger precursor while offering advantages in resource utilization and inference speed. This makes it a compelling choice for applications constrained by computational resources or requiring expedited processing times [119], [109].

- **DistilBERT tokenizer**

The DistilBERT tokenizer is designed to preprocess text data for input into the DistilBERT model or other classifier. It segments the input text into individual tokens, each representing a discrete unit of meaning, such as words or subwords. Additionally, it applies various text normalization techniques, including lowercasing, punctuation handling, and special tokenization rules specific to the DistilBERT model. The tokenizer facilitates the conversion of raw text data into a format compatible with DistilBERT's or DL model input requirements, enabling efficient processing and analysis of natural language text [74].
The tokenizer for DistilBERT employs WordPiece subword segmentation. This tokenizer converts raw text into integer sequences using the
'keras_nlp.tokenizers.WordPieceTokenizer'. Notably, it encompasses functionality to ensure the inclusion of all essential special tokens required by DistilBERT models. Additionally, it offers a convenient 'from_preset()' method, facilitating the automatic retrieval of a compatible vocabulary tailored for specific DistilBERT presets [74].
The utilized datasets comprised only a single data column containing URL text. NLP is employed in this data column for feature extraction. As illustrated in Figure 5.1, the feature extraction stage consists of two parts: applying NLP techniques to the URL input and extracting URL features. Subsequently, two scenarios are tested in our study. Firstly, solely utilizing the output of the first part as input to the deep learning classification model and secondly, incorporating the features extracted from the URL into the initial input as input to the classification model. The output of the features extraction stage comprises either word embeddings or a combination of word embeddings and extracted URL features [109].

5.2.3.3 Steps for Applying Natural Language Processing to Input URLs

This subsection delineates the steps involved in applying natural language processing to the datasets for feature engineering. Fig5.2 presents a detailed flowchart illustrating the application of the transformer model to URL text [109].

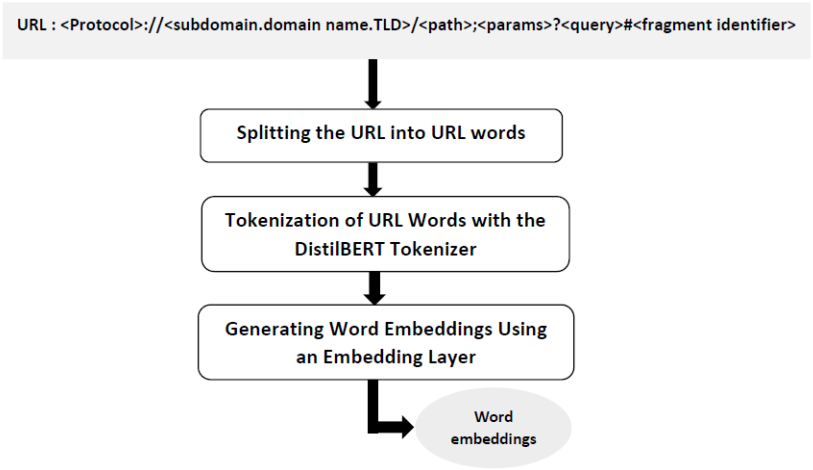


Figure 5.2: A detailed flowchart of features extraction from URLs using DistilBERT [109].

1. **Parsing the URL**
   First, the URL column of the datasets is parsed and split into a column called "splitted_URL," which contains each part of the URL, as depicted in Figure4.2. The URL format includes the following parts:
   <Protocol>://<subdomain.domain-name.TLD>/<path>;<params>?<query>#<fragment identifier>

   - Protocol (Scheme): Defines the protocol used to access the resource, such as HTTP or HTTPS.
   - Sub-domain: A subdivision of the domain name is often used to specify a particular server or service within an organization's domain.
   - Domain name: The main part of a URL that identifies the website. For example, (in Figure 4.2)"www.example.com", "example" is the domain name.
   - Top level domain (TLD): The highest level in the domain hierarchy indicates the type or purpose of the domain, such as .com, .org, .net, etc.
   - Path: Specifies the location of a specific resource or page on the web server.
   - Query string: A set of parameters passed to a web page, typically used for dynamic content generation or querying a database (optional).
   - Fragment Identifier: A portion of the URL that specifies a specific section within a resource, usually denoted by a "#", commonly used in linking to specific sections of a webpage (optional).

   After splitting the URL, we remove a set of symbols and non-text characters [d/.//,_˜%-+=@], leaving only the text words. This results in a list of substrings that contain only the text words from the original URL (see Figure 5.3).

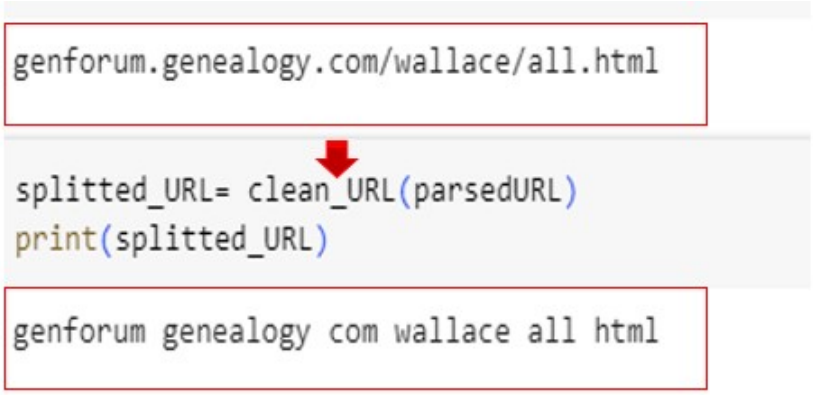


Figure 5.3: Illustration of splitting a URL to extract URL words from an example URL string.

2. **Tokenization of URL Words Using DistilBERT Tokenizer**
   The words derived from URL strings are introduced as inputs for DistilBERT tokenization. Unlike conventional character-level tokenization approaches, where identical characters receive identical embeddings, DistilBERT tokenization takes into account the significance of letters across diverse vocabularies. This enables it to produce more meaningful representations of the text compared to methods that treat each character uniformly. In this investigation, the tokenizer from the distillery-base-uncased model was utilized for tokenizing the URL word strings [109]. Figure 5.4 illustrates an example of URL tokenization.
   In this example, the tokenizer has segmented the input text "genforum genealogy com wallace all html" into 10 tokens. This segmentation occurs due to the inclusion of subword units representing parts of words, as well as the addition of special tokens like [CLS] (classification) and [SEP] (separator) tokens by the tokenizer. The tokenizer's WordPiece tokenization strategy is responsible for generating these tokens, breaking down words into subword units as required [109].

```
# Initialize DistilBERT tokenizer
tokenizer = DistilBertTokenizer.from_pretrained('distilbert-base-uncased')

print(splitted_URL)

genforum genealogy com wallace all html

# Tokenize the text
tokenized_text = tokenizer.encode(splitted_URL, add_special_tokens=True)

print(tokenized_text)

[101, 8991, 29278, 2819, 26684, 4012, 7825, 2035, 16129, 102]
```

Figure 5.4: Illustration of DistilBERT tokenization for an example URL words string.

3. **Padding the tokenized text and generating word embeddings with an Embedding layer** In this step, the generated sequence is padded to a maximum length of 32 using TensorFlow's 'pad_sequences' function. The vocabulary size is set based on the tokenizer's vocabulary, and determined by the number of unique tokens in the tokenizer.vocab "vocab size = len(tokenizer.vocab)". Next, we set up an embedding layer in Keras, where the input dimension matches the vocabulary size and the output dimension is set to 50. This process results in embedding words that serve as input to the classification model [109].

5.2.3.4 Extracting Features from URLs

To enhance classification performance and facilitate comparison between the two scenarios, we extract 21 of the most common features from the URL text across both datasets. The selection of these features is based on extensive experimentation, and the number of features is determined by identifying those that yield the best results. We chose a list of the most effective features based on our experience in the field and our previous works [107], [108]. Then, we added one feature to the first scenario each time until we reached the number that yielded the best results, which is 21. These extracted features are then concatenated with the word embeddings to serve as input to the classification model in the second scenario. Table 5.2 summarizes the extracted features [109].

Table 5.2: Description of the extracted URL features [109].

Feature	Description	Feature Type
use_of_ip	It verifies whether an IP address is substituted for a domain name in the URL. This is a frequently employed tactic by phishers to mask their identities.	Binary (1 in case of phishing and 0 in case of legitimate)
abnormal_url	It is derived from the WHOIS database. Benign websites typically maintain a consistent identity within their URLs, so any deviation from this consistency could suggest a potentially phishing website.	Binary
short_url	This feature detects URLs that utilize URL shortening systems such as bit.ly, go2l.ink, or goo.gl, as these tricks are frequently employed by attackers to obscure the actual destination of the URL.	Binary
sus_url	it checks the presence of frequently used suspicious words in the URL, such as "lucky", "login", "service", "bonus", "free", and "bank". URL containing these words is deemed suspicious.	binary

Continued on next page

Table 5.2 – Continued from previous page

Feature	Description	Feature Type
count.	This feature determines the number of subdomains present in a URL, as phishing websites often employ multiple subdomains separated by dots. A URL containing more than three dots is regarded as having an increased likelihood of being a phishing website.	Numeric
count-www	This feature recognizes URLs with either no "www" subdomain or multiple instances of "www", since secure websites typically have only one "www" in their URLs.	Numeric
count@	This feature tallies the occurrences of the "@" symbol in the URL, which usually signifies the beginning of an email address and can be utilized to disregard everything preceding it.	Numeric
count_dir	This feature searches for the presence of numerous directories in the URL, as this frequently indicates a suspicious website.	Numeric
count_embed_domian	This feature counts the occurrence of "//" in the URL to examine the number of embedded domains in the URL.	Numeric

Continued on next page

Table 5.2 – Continued from previous page

Feature	Description	Feature Type
count-https	This feature counts the occurrence of “HTTPS” protocol in the URL, as most malicious URLs tend not to use HTTPS.	Numeric
count-http	This feature tallies the occurrences of "HTTP" in the URL, since phishing websites often feature multiple instances of "HTTP" in their URL.	Numeric
count%	This function calculates the number of percent "%" symbols in the URL, since malicious URLs typically contain more spaces in their URL than benign sites.	Numeric
count?	This feature calculates the number of "?" symbols in the URLs, which indicates a query string holding data to be transferred to the server.	Numeric
count-	It calculates the number of dashes in the URL, since phishers often incorporate them into the suffix or prefix of a brand name to give it a false sense of authenticity.	Numeric
count=	This feature calculates the number of equal symbol in the URL, which signifies the transmission of variable values from one page to another. URL containing more "=" symbols is deemed to be suspicious.	Numeric

Continued on next page

Table 5.2 – Continued from previous page

Feature	Description	Feature Type
url_length	This feature calculates the length of the URL. A long URL is often indicative of malicious intent, as it is frequently used to obscure the domain name.	Numeric
hostname_length	It examines the size of the hostname in the URL. since this is a crucial indicator of malicious websites.	Numeric
fd_length	This feature calculates the length of the first directory in the URL, since it is a sign of malicious websites.	Numeric
tld_length	This feature analyzes the size of the Top Level Domain in the URL. Attackers frequently exploit long TLDs to craft fake domains that closely resemble legitimate ones.	Numeric
count-digits	This feature counts the number of digits in the URL. A high count of digits in the URL is often indicative of malicious intent.	Numeric
count-letters	It counts the number of letters in the URL.	Numeric

5.2.4 Classification Stage

For the classification stage, we developed a supervised classification hybrid deep learning algorithm specifically tailored to handle both word embeddings and URL features. This hybrid algorithm combines convolutional neural network (CNN) and Long Short-Term Memory (LSTM) models.

CNN is a neural-based approach that applies a feature function to constituent words or n-grams to extract higher-level features. It is commonly used in text classification tasks because it effectively captures low-level features and controls the length of dependency. LSTM, short for Long Short-

Term Memory, is a type of recurrent neural network (RNN) that is capable of learning long-term dependencies in sequence data. It is particularly well-suited for tasks involving sequential data due to its ability to retain information over long sequences.

The decision to utilize CNN and LSTM models stems from their promising performance in the literature and their recognized effectiveness in detecting phishing websites. Combining them in a hybrid model leverages the strengths of both models and overcomes the limitations of each, resulting in a more comprehensive approach to classification. This hybrid model architecture combines the strengths of both CNN and LSTM models to effectively capture spatial and sequential information from the input data, resulting in improved performance for detecting malicious websites. Figure5.5 illustrates the learning process flow of the transformer learning and the deep learning model for detecting malicious websites, along with a detailed architecture of the employed classification model [109].

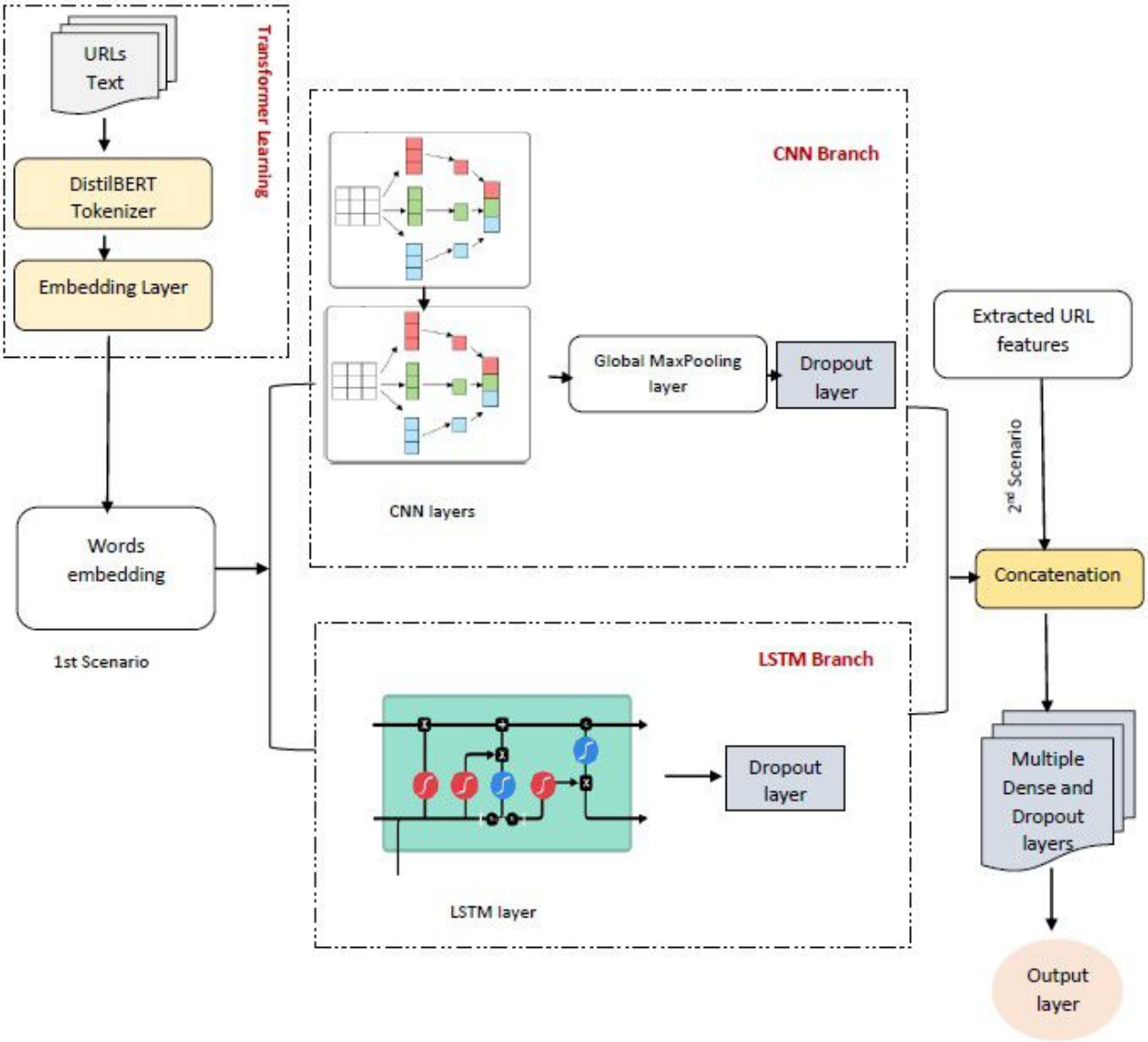


Figure 5.5: Architecture of the proposed classification model [109].

The proposed hybrid DL model architecture consists of several interconnected layers [109]:

- **Embedding Layer:** DistilBERT is employed to tokenize the URL input, and an Embedding layer is initialized with the specified vocabulary size and embedding dimension (output_dim=50). It converts the input text into dense word embeddings that serve as the primary input matrix for subsequent layers.

- **CNN layers:** This branch utilizes Convolutional Neural Network (CNN) layers with max pooling and dropout. The Conv1D layers with activation functions extract feature maps

from the input embeddings. Two Conv1D layers with different filter sizes (32 and 64) and relu activation functions are applied sequentially to capture different levels of abstraction. The output is then processed by a GlobalMaxPooling1D layer to extract the most salient features.

- **Max Pooling and Dropout:** GlobalMaxPooling1D layer is used to pool the maximum value of each feature map. Dropout regularization with a rate of 0.5 is applied to mitigate overfitting.

- **LSTM Branch:**This branch employs a Long Short-Term Memory (LSTM) layer with dropout regularization. The LSTM layer processes the input embeddings to capture sequential dependencies within the data.

- **Concatenation:**The output from the CNN and LSTM branches is concatenated into a single tensor in the first scenario and with additional binary and statistical inputs in the second scenario. This step combines the learned features from the text data with additional information.

- **Dense Layers:** The concatenated features are then passed through multiple dense layers with relu activation functions, dropout regularization, and L2 kernel regularization. These layers further process and refine the feature representations learned from the input data.

- **Output Layer:** The final dense layer with a sigmoid activation function outputs the probability of the input being classified as malicious or benign.

We utilize the Adam optimizer and the binary cross-entropy loss function to compile the model. Each parameter used during the design of the hybrid model is selected after a fine-tuning process to ensure optimal performance. The summary of the model architecture, including the layers and parameters, is depicted in Figures 5.6 and 5.7 for the two scenarios: one using only the input text (words embeddings), and the other using words embeddings along with URL features (binary and statistical) as input to the model, respectively [109].

Model: "functional_3"

Layer (type)	Output Shape	Param #	Connected to
text_input (InputLayer)	(None, 32)	0	-
embedding_1 (Embedding)	(None, 32, 50)	1,526,100	text_input[0][0]
conv1d_2 (Conv1D)	(None, 31, 32)	3,232	embedding_1[0][0]
conv1d_3 (Conv1D)	(None, 30, 64)	4,160	conv1d_2[0][0]
global_max_pooling1d_1 (GlobalMaxPooling1D)	(None, 64)	0	conv1d_3[0][0]
lstm_1 (LSTM)	(None, 64)	29,440	embedding_1[0][0]
dropout_5 (Dropout)	(None, 64)	0	global_max_pooling1d_…
dropout_6 (Dropout)	(None, 64)	0	lstm_1[0][0]
concatenate_1 (Concatenate)	(None, 128)	0	dropout_5[0][0], dropout_6[0][0]
dense_4 (Dense)	(None, 128)	16,512	concatenate_1[0][0]
dropout_7 (Dropout)	(None, 128)	0	dense_4[0][0]
dense_5 (Dense)	(None, 128)	16,512	dropout_7[0][0]
dropout_8 (Dropout)	(None, 128)	0	dense_5[0][0]
dense_6 (Dense)	(None, 128)	16,512	dropout_8[0][0]
dropout_9 (Dropout)	(None, 128)	0	dense_6[0][0]
input_layer (InputLayer)	(None, 32)	0	-
dense_7 (Dense)	(None, 1)	129	dropout_9[0][0]

Total params: 1,612,597 (6.15 MB)
Trainable params: 1,612,597 (6.15 MB)
Non-trainable params: 0 (0.00 B)

Figure 5.6: Summary of the CNN-LSTM model when using only word embeddings as input [109].

Model: "functional_1"

Layer (type)	Output Shape	Param #	Connected to
text_input (InputLayer)	(None, 32)	0	-
embedding_1 (Embedding)	(None, 32, 50)	1,526,100	text_input[0][0]
conv1d (Conv1D)	(None, 31, 32)	3,232	embedding_1[0][0]
conv1d_1 (Conv1D)	(None, 30, 64)	4,160	conv1d[0][0]
global_max_pooling1d (GlobalMaxPooling1D)	(None, 64)	0	conv1d_1[0][0]
lstm (LSTM)	(None, 32)	10,624	embedding_1[0][0]
dropout (Dropout)	(None, 64)	0	global_max_pooling1d[…
dropout_1 (Dropout)	(None, 32)	0	lstm[0][0]
binary_input (InputLayer)	(None, 4)	0	-
statistical_input (InputLayer)	(None, 17)	0	-
concatenate (Concatenate)	(None, 117)	0	dropout[0][0], dropout_1[0][0], binary_input[0][0], statistical_input[0][…
dense (Dense)	(None, 128)	15,104	concatenate[0][0]
dropout_2 (Dropout)	(None, 128)	0	dense[0][0]
dense_1 (Dense)	(None, 128)	16,512	dropout_2[0][0]
dropout_3 (Dropout)	(None, 128)	0	dense_1[0][0]
dense_2 (Dense)	(None, 128)	16,512	dropout_3[0][0]
dropout_4 (Dropout)	(None, 128)	0	dense_2[0][0]
dense_3 (Dense)	(None, 1)	129	dropout_4[0][0]

Total params: 1,592,373 (6.07 MB)
Trainable params: 1,592,373 (6.07 MB)
Non-trainable params: 0 (0.00 B)

Figure 5.7: Summary of the CNN-LSTM model when using word embeddings + extracted URL features as input [109].

5.3 Implementation and Evaluation of performances

This study conducted experiments on Google Colab using the Python programming language, along with the TensorFlow (version 2.18.0) and Keras (version 3.5.0) packages, utilizing two distinct public datasets obtained from Kaggle (Table 5.1). Each dataset was split into training and testing sets in an 80:20 ratio, respectively. Subsequently, our proposed hybrid deep learning (DL) model was fitted to each dataset under two scenarios:

- Firstly, utilizing only text input (word embeddings).
- Secondly, incorporating text input along with the 21 extracted URL features (binary and statistical features Table 5.2).

The classification model was trained using a balanced training dataset and tested on the test dataset over 20 epochs for each case. A callback function with early stopping was implemented to halt the training process when the validation accuracy stopped improving for a predefined number of epochs, thereby preventing overfitting. Detailed parameters for each scenario in the experiments are outlined in Figures 5.6 and 5.7 [109].

5.3.1 Results obtained

To evaluate the proposed approach, we utilized the performance metrics outlined in (subsection 4.2.1.4). The performance of the proposed approach, employing the CNN-LSTM model with data 1 (3), under two scenarios, is presented in Table 5.3. Figures 5.8 and 5.9 present the confusion matrices for the two respective scenarios using Data 1 and Data2.
The results presented in Table 5.3 indicate that the proposed approach exhibits a strong performance in classifying malicious URLs, achieving scores that exceed 97% for the accuracy metric, demonstrating the effectiveness of the proposed method. Moreover, we observed improvements in almost all performance metrics in the second scenario, with accuracy reaching 97.35% with the FPR value reduced to 1.66%. This suggests that incorporating the extracted URL features significantly enhances the detection capabilities of the system.

Similarly, Table 5.4 outlines the performance of the proposed approach, employing the CNN-LSTM model with data 2 (4), under two scenarios. From the results displayed in Table 5.4, it is evident that the proposed approach demonstrates strong performance in classifying malicious URLs achieving scores exceeding 96.0% for the accuracy metric. Furthermore, we observe improvements in all performance measures in the second scenario, with an accuracy of 98.19%, a precision of 98.22%, a recall of 96.32%, and an F1 score of 97.26% with the FPR value reduced to 0.87%. This further demonstrates that including the extracted URL features significantly improves the detection performance [109].

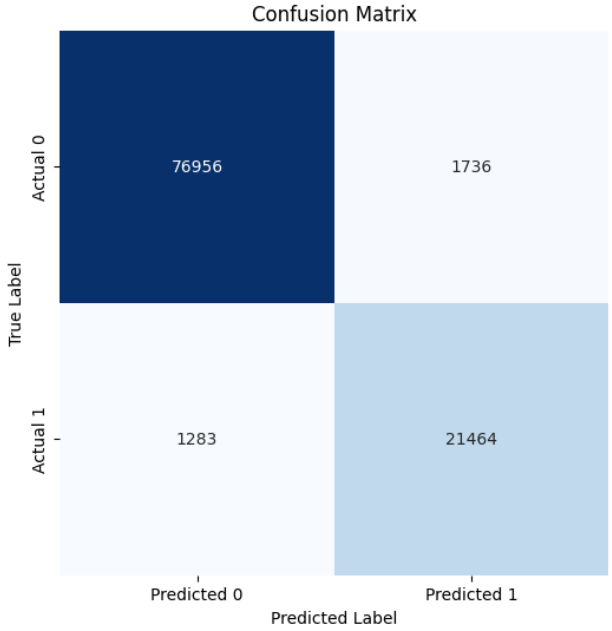
The improvements in the second scenario are attributed to the incorporation of additional URL features, which provide a multi-dimensional view by combining semantic (word embeddings) and structural insights. This holistic approach enhances the CNN-LSTM model's ability to detect subtle malicious patterns, such as obfuscation, phishing tactics, and suspicious domain characteristics. Features (presented in Table 5.2) such as URL length, number of subdomains, and special characters offer critical insights into attackers' tactics. These features complement word embeddings by providing a more comprehensive understanding of URLs, helping to detect URLs that may appear benign textually but exhibit suspicious structural traits, thus enhancing accuracy and reducing false negatives [109].

Table 5.3: Results obtained by the proposed approach across the two scenarios on data1 [109].

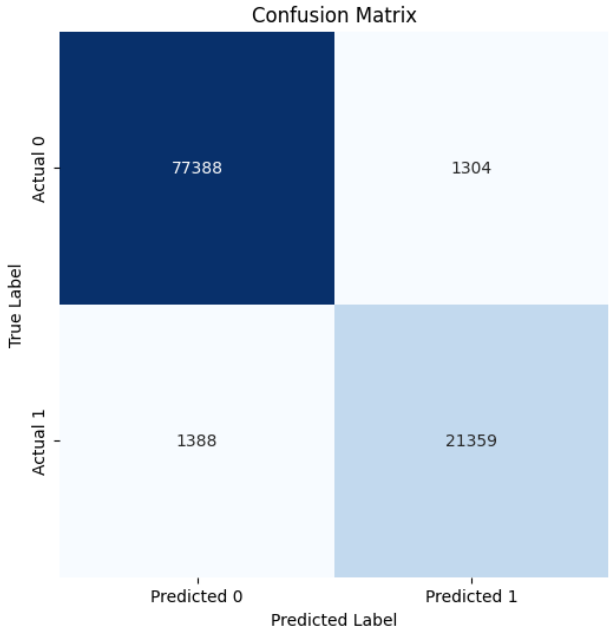
	Accuracy(%)	**Precision(%)**	**Recall(%)**	**F1-score(%)**	**FPR(%)**
Scenario1 : text input(word embedding)	97.02	92.52	94.36	93.43	2.21
Scenario2:text input(word embedding)+ 21 extracted URL features	97.35	94.25	93.90	94.07	1.66

Table 5.4: Results obtained by the proposed approach across the two scenarios on data2 [109].

	Accuracy(%)	**Precision(%)**	**Recall(%)**	**F1-score(%)**	**FPR(%)**
Scenario1 : text input(word embedding)	96.04	95.16	92.80	93.97	2.35
Scenario2:text input(word embedding)+ 21 extracted URL features	98.19	98.22	96.32	97.26	0.87

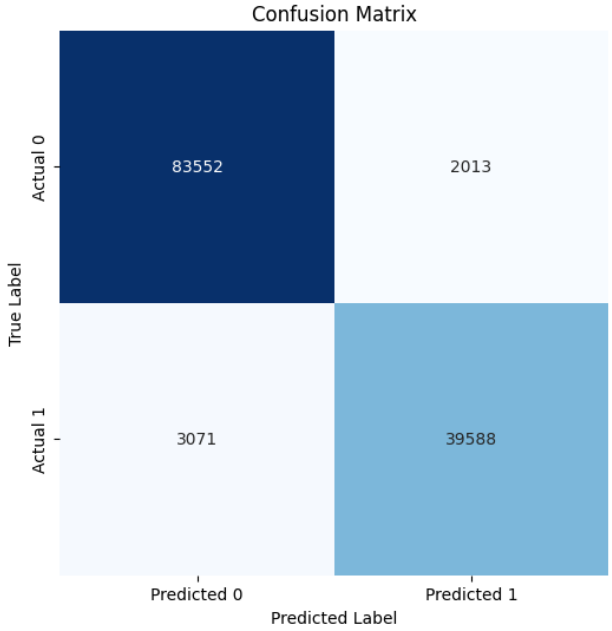


(a) First scenario

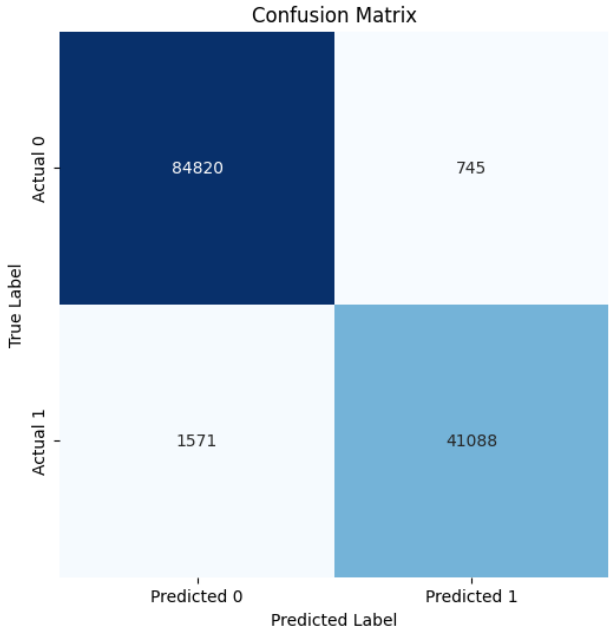


(b) Second scenario

Figure 5.8: Confusion matrices of the CNN-LSTM model using data 1 [109].



(a) First scenario



(b) Second scenario

Figure 5.9: Confusion matrices of the CNN-LSTM model using data 2 [109].

5.3.2 Comparison and Summary

In this section, we compare the performance of our proposed approach with that of relevant literature studies.
Tables 5.5 and 5.6 present the results obtained by our proposed method across the two datasets (data1 and data2), respectively, alongside results from previous studies. These studies were selected for comparison because they all utilize URLs for phishing detection, employing various methods on the same datasets as ours. Figures 5.10 and 5.11 display the outcomes of our proposed method on data1 and data2, respectively, alongside results from earlier investigations.
For our approach using data 1, as shown in Table 5.5, our method outperforms [49] in terms of accuracy, which used the same dataset, achieving the highest accuracy of 97.35%, close to the accuracy achieved by [78]. Additionally, our approach surpasses the study in [49] in terms of recall across both scenarios and in terms of the F1-score in Scenario 2.

For our approach using data 2, as indicated in Table 5.6, our method in the second scenario surpasses all the metric values reported by [69], achieving the highest accuracy of 98.19%. It also achieved a precision of 98.22%, a recall of 96.32%, an F1-score of 97.26%, and a lower false positive rate (FPR) of 0.87%. Additionally, we compared our results with those reported in [63] for precision, recall, and F1-score, as their performance evaluation did not include accuracy or FPR. Notably, the metrics reported by our approach using Scenario 2 are almost comparable to theirs.
In summary, the proposed approach demonstrated excellent performance in detecting malicious websites under various conditions (two scenarios) and utilizing two large datasets, highlighting the model's generalizability. Furthermore, it outperformed some relevant studies in the literature that used the same datasets, confirming the effectiveness of the malicious website detection system. However, it requires further improvements in the future to enhance results and reduce both false positive (FP) and false negative (FN) values [109].

5.4 Conclusion

In this chapter, we have proposed an enhanced approach to detect malicious URLs by combining two emerging technologies: Transformer Learning and Deep Learning. We employed DistilBERT for feature extraction and a hybrid DL model, CNN-LSTM, for classifying URLs as either malicious or benign. To evaluate our proposed approach, we utilized two large public datasets from Kaggle (named data1 [125] and data2 [118]). Each dataset was applied in two scenarios: one using only text input (word embeddings), and the other incorporating additional binary and statistical features. This allowed us to assess the performance of our approach across various conditions. The evaluation of performance revealed excellent results across different conditions, particularly in scenario 2, where the addition of URL features led to the highest accuracy value of 98.19% on data1. Furthermore, our approach outperformed most related studies, demonstrating that the

Table 5.5: Comparison of results achieved by our approach with relevant studies in the literature using data 1 [109].

	Our approach on 1st scenario	**Our approach on 2nd scenario**	**Elsadig et al. [49] (2022)**	**Lakshmanarao et al.[78] (2021)**
Accuracy (%)	97.02	97.35	96.66	97.5
Precision (%)	92.52	94.25	96.66	/
Recall(%)	94.36	93.90	83.43	/
F1-Score(%)	93.43	94.07	93.63	/
FPR(%)	2.21	1.66	0.7	/
Classifier used	CNN-LSTM	CNN-LSTM	CNN	Logistic Regression, K-NN, Decision Tree, Random Forest
Feature extraction method	NLP (transfer learning) DistilBERT	NLP (transfer learning) DistilBERT + 21 URL features	BERT+21 features	count vectorizer, hashing vectorizer with IDF vectorizer

Table 5.6: Comparison of results achieved by our approach with relevant studies in the literature using data 2 [109].

	Our approach on 1st scenario	**Our approach on 2nd scenario**	**Hoang et al. [63](2023)**	**Sajadul Islam et al. [69](2023)**
Accuracy (%)	96.04	98.19	/	97.0
Precision (%)	95.16	98.22	99.26	96
Recall (%)	92.80	96.32	98.73	95
F1-Score (%)	93.97	97.26	98.99	95
FPR(%)	2.35	0.87	/	1.83
Classifier used	CNN-LSTM	CNN-LSTM	CNN	Random Forest, LightGBM, and XGBoost
Feature extraction method	NLP (transfer learning) DistilBERT	NLP (transfer learning) DistilBERT + 21 URL features	3 extracted features+ embedding layer	Features extraction manually from website content and metadata.

use of DistilBERT and DL significantly enhanced detection performance, affirming the successful operation of our proposed mechanism.

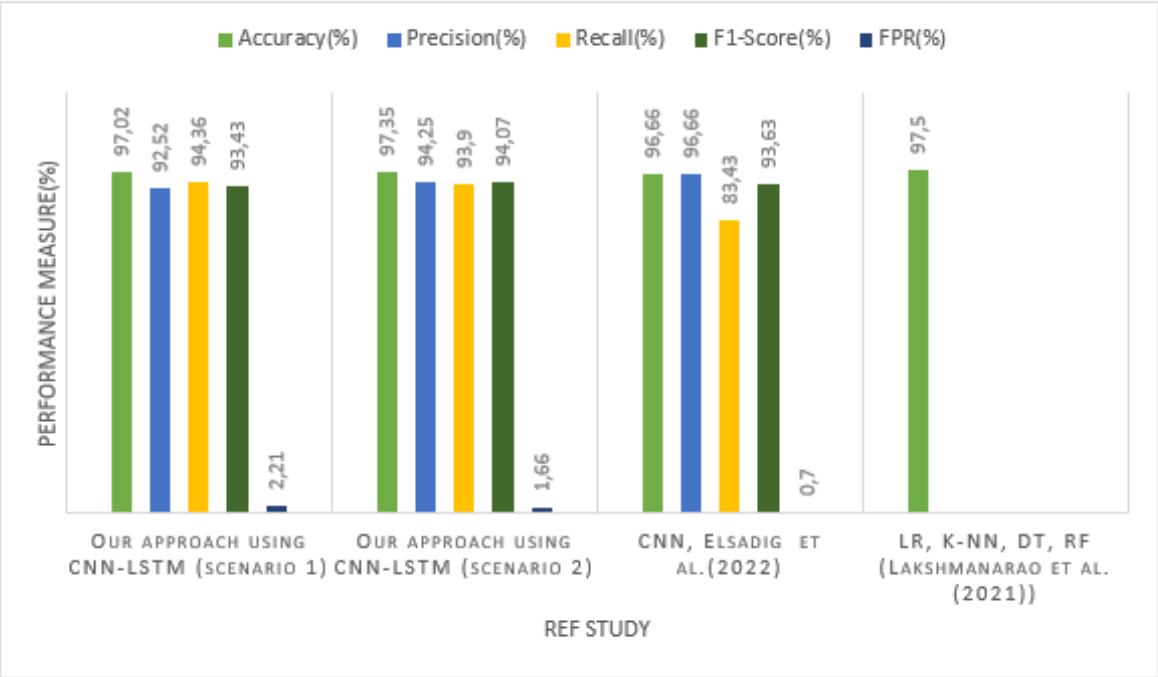


Figure 5.10: Comparison with previous studies based on data1 [109].

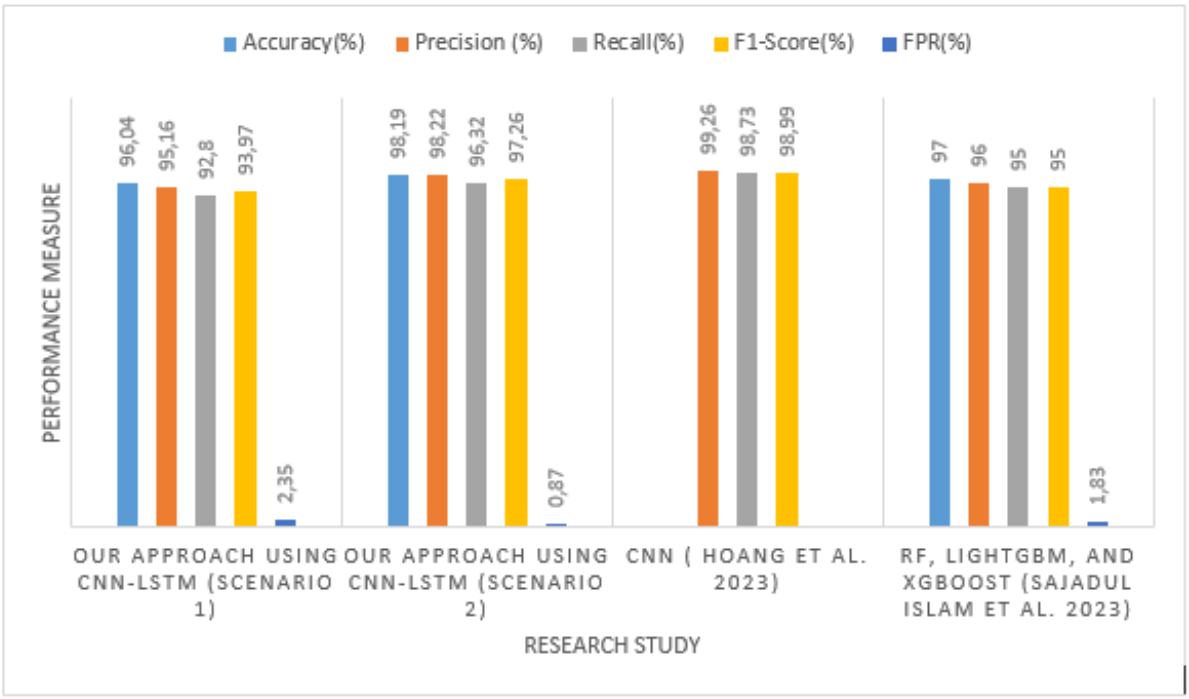


Figure 5.11: Comparison with previous studies based on data 2 [109].

Chapter. 6

General Conclusion

Nowadays, with the meteoric rise of online activities, web users' privacy has become at risk. The protection of privacy has become one of the major concerns for Internet users. In this thesis, we focus on threats that pose a significant risk to privacy on the web, specifically cyberattacks. Among these, phishing attacks are the most prevalent in the field of cybersecurity. The website phishing attack is the most widespread in the field of cybersecurity. Despite the anti-phishing solutions proposed, this attack continues to grow in several sectors such as e-banking, online payment, e-commerce, etc., because attackers continue to invent new tricks to overcome various detection systems. Furthermore, existing solutions, particularly those based on conventional methods, suffer from several limitations on multiple fronts or do not guarantee good performance. To address these problems, the objective of this thesis is to focus on studying the feasibility and implementation of an approach for detecting phishing websites based on emerging, lightweight, and robust technologies, allowing for effective detection of this attack to contribute to cybersecurity and protect web user privacy.

In the first stage, we began with an overview of online privacy and cybersecurity. The objective was to define the basic concepts of this domain to better understand it. Additionally, we presented a literature review in the field of phishing to narrow down the scope of our work and understand it before being able to delve into the contributions introduced in this thesis.

The contributions of this thesis focus on cybersecurity to help protect users' privacy on the web by addressing the issue of phishing attacks. This is achieved by introducing intelligent systems for highly efficient and high-performance phishing website detection, based on emerging technologies that adapt to new phishing attacks.
In our first contribution, we proposed a deep learning-based approach for detecting phishing websites using both 1D CNN and 2D CNN architectures. This approach incorporates three features:

URL-based, content-based, and third-party services-based. We conducted a parameter tuning process, performing multiple experiments to optimize the parameters of the CNN model. Upon comparing our proposed solution with other relevant works, the tests conducted on the proposed system demonstrated improvement over existing solutions.
Our second contribution introduces a robust mechanism for detecting phishing websites, employing multiple distinct classifiers leveraging both machine learning (ML) and deep learning (DL), including hybrid DL models. The proposed approach addresses the challenge of data imbalance through the utilization of the SMOTE-Tomek link method, while also benefiting from feature selection using the permutation importance method. We evaluate the performance of our proposed mechanism using two variants of datasets under different conditions. Experimental results demonstrate that the application of feature selection and data balancing techniques significantly enhances the efficacy of the phishing detection mechanism across all employed classifiers, rendering it more robust. Comparison with existing literature indicates that our mechanism outperforms baseline models across various metrics. These results affirm the successful operation of our proposed mechanism in detecting phishing websites.
In the third contribution, we presented an improved method for detecting malicious URLs by integrating two emerging technologies: Natural Language Processing (NLP) and Deep Learning (DL), to enhance performance. For feature extraction, we employed DistilBERT, and for classification, we utilized a hybrid DL model, CNN-LSTM. We assessed the performance of our proposed approach across various scenarios, using two extensive public datasets. The results demonstrated outstanding performance, achieving a detection accuracy of 98.19%, thereby affirming its effectiveness in identifying malicious URLs. Moreover, our approach surpassed the majority of related studies, underscoring the significant enhancement in detection performance facilitated by the use of DistilBERT and DL.

Finally, we implemented a web application to detect phishing websites, aiming to enhance usability for web users by providing real-time protection. This web application underwent testing using the approach described in the second contribution, but it can be adapted for each mechanism proposed in this thesis.

Future Works

In future work, our objectives are to:

- Enhance the proposed phishing detection mechanisms by incorporating additional preprocessing techniques, including various feature selection algorithms such as the SHAP method (SHapley Additive exPlanations) when computational resources allow. Additionally, we plan to use additional datasets to ensure the adaptability of the proposed systems and improve detection performance.

- Addressing the proposed mechanism's limitations by improving its performance to reduce FP and FN rates and updating the models with new features to detect advanced phishing tactics.

- Deploy our phishing detection web application as a secure browser extension prediction system to elevate its functionality and presentation, delivering instantaneous and automated safeguarding for end-users.

- Address other types of phishing attacks such as phishing emails, phishing attacks based on images and voices, as well as other cybersecurity attacks such as man-in-the-middle attacks, DoS attacks, etc.

- Address more challenges in the online privacy field, such as protecting personal data on the web using cryptographic techniques.

References

[1] Ruba Abu Khurma, Khair Eddin Sabri, Pedro Castillo, and Ibrahim Aljarah. *Salp Swarm Optimization Search Based Feature Selection for Enhanced Phishing Websites Detection*, pages 146–161. 04 2021. ISBN 978-3-030-72698-0. doi: 10.1007/978-3-030-72699-7_10.

[2] M.A. Adebowale, K.T. Lwin, and M.A. Hossain. Intelligent phishing detection scheme using deep learning algorithms. *Journal of Enterprise Information Management*, ahead-of-print(ahead-of-print), 2020. doi: 10.1108/JEIM-01-2020-0036. URL `https://doi.org/10.1108/JEIM-01-2020-0036`.

[3] P. Agrawal, H. F. Abutarboush, T. Ganesh, and A. W. Mohamed. Metaheuristic algorithms on feature selection: a survey of one decade of research (2009–2019). *IEEE Access*, 9: 26766–26791, 2021.

[4] Bruno Aguirre. Fraud in disaster: rethinking the phases. *Int. J. Disaster Risk Reduct*, 39: 101232, 2019. doi: 10.1016/j.ijdrr.2019.101232.

[5] S. Al-Ahmadi and Y. Alharbi. A deep learning technique for web phishing detection combined url features and visual similarity. *International Journal of Computer Networks and Communications (IJCNC)*, 12(5), 2020.

[6] Y. Al-Hamar, H. Kolivand, and A. Al-Hamar. Phishing attacks in qatar: A literature review of the problems and solutions. In *2019 12th International Conference on Developments in eSystems Engineering (DeSE)*, 2019.

[7] Ahmed Aljofey, Qingqing Jiang, Qing Qu, Mengting Huang, and Jean Pierre Niyigena. An effective phishing detection model based on character-level convolutional neural network from URL. *Electronics*, 9(9):1514, 2020.

[8] N. Aloysius and M. Geetha. A review on deep convolutional neural networks. In *2017 International Conference on Communication and Signal Processing (ICCSP)*, pages 0588–0592, Chennai, India, 2017. doi: 10.1109/ICCSP.2017.8286426.

[9] Z. Alshingiti, R. Alaqel, J. Al-Muhtadi, Q.E.U. Haq, K. Saleem, and M.H. Faheem. A deep learning-based phishing detection system using cnn, lstm, and lstm-cnn. *Electronics*, 12:232, 2023. doi: 10.3390/electronics12010232. URL `https://doi.org/10.3390/electronics12010232`.

[10] Anti-Phishing Working Group (APWG). Trend reports. 1st quarter 2020 plus covid-19 coverage technical report. Technical report, 2020. Accessed on March 2024.

[11] Anti-Phishing Working Group (APWG). Phishing activity trend reports. 3rd quarter 2020 technical report. Technical report, 2020. Accessed on March 2024.

[12] Anti-Phishing Working Group (APWG). Trend reports. 1st quarter 2022 technical report. Technical report, 2022. Accessed on March 2024.

[13] Anti-Phishing Working Group (APWG). Trend reports. summary – 2nd quarter 2023. Technical report, 2023. Accessed on: December 2023.

[14] APWG. 4th quarter 2022: Phishing activity trends report. Anti-Phishing Working Group, 2022. URL `https://apwg.org/trendsreports/`.

[15] S. Ariyadasa, S. Fernando, and S. Fernando. Detecting phishing attacks using a combined model of lstm and cnn. *International Journal of Advanced and Applied Sciences*, 7(7): 56–67, 2020.

[16] Abid All Awan. The machine learning life cycle explained. `https://www.datacamp.com/blog/machine-learning-lifecycle-explained`, 2022. Last updated: October 2022.

[17] Sai Balaji. Binary image classifier cnn using tensorflow. `https://medium.com/techiepedia/binary-image-classifier-cnn-using-tensorflow-a3f5d6746697`. Last updated: August 2020.

[18] A. Basit, M. Zafar, X. Liu, and et al. A comprehensive survey of ai-enabled phishing attacks detection techniques. *Telecommunication Systems*, 76:139–154, 2021. doi: 10.1007/s11235-020-00733-2. URL `https://doi.org/10.1007/s11235-020-00733-2`.

[19] A Belabed. *La protection de la vie privée sur Internet*. PhD thesis, University of Abou Bekr Belkaid Tlemcen UABT, 2018.

[20] Alexy Bhowmick and Shyamanta M. Hazarika. E-mail spam filtering: A review of techniques and trends. 2018. URL https://api.semanticscholar.org/CorpusID:196045321.

[21] T Bienkowski. What is a Packet Sniffer? . https://www.netscout.com/what-is/sniffer.

[22] bitdefender. What is online privacy? and why is it important? https://www.bitdefender.com/cyberpedia/what-is-online-privacy/. (accessed: 01.05.2023).

[23] Bitdefender.com. What is online privacy? and why is it important? https://www.bitdefender.com/cyberpedia/what-is-online-privacy/. Accessed on: December 2023.

[24] Lucija Brezočnik, Iztok Fister jr, and Grega Vrbančič. *Applying Differential Evolution with Threshold Mechanism for Feature Selection on a Phishing Websites Classification*, pages 11–18. 09 2019. ISBN 978-3-030-30277-1. doi: 10.1007/978-3-030-30278-8_2.

[25] J. Brownlee. Use early stopping to halt the training of neural networks at the right time. https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/, . Last updated: Dec 10, 2018.

[26] J. Brownlee. How to configure the learning rate when training deep learning neural networks. https://machinelearningmastery.com/learning-rate-for-deep-learning-neural-networks/, . Last updated: Jan 23, 2019.

[27] J. Brownlee. Understand the impact of learning rate on neural network performance. https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/, . Last updated: Jan 25, 2019.

[28] camerecole.org. Description des concepts de base de la sécurité informatique. https://www.camerecole.org/classes/1155-description-des-concepts-de-base-de-la-securite-informatiques.html. Accessed January 2024.

[29] S. Chanti and T. Chithralekha. Classification of anti phishing solutions. *SN Computer Science*, 2019. URL https://doi.org/10.1007/s42979-019-0011-2.

[30] Mohamed Chawki. Essai sur la notion de cybercriminalité. IEHEI, July 2006.

[31] CS Chen, SA Su, and YC Hung. Protecting computer users from online frauds. Google Patents, 2011.

[32] J. Chen, E. Haber, R. Kang, G. Hsieh, and J Mahmud. Making use of derived personality: The case of social media ad targeting. *Proceedings of the International AAAI Conference on Web and Social Media*, 9(1):51–60, 2021. doi: 10.1609/icwsm.v9i1.14599.

[33] T-C Chen, T Stepan, S Dick, and J Miller. An anti-phishing system employing diffused information. *Journal of ACM Transactions on Information and System Security (TISSEC)*, 16(4):1–31, 2014.

[34] Wenwu Chen, Wei Zhang, and Yang Su. Phishing detection research based on lstm recurrent neural network. In *4th International Conference of Pioneering Computer Scientists, Engineers and Educators, ICPCSEE 2018, Zhengzhou, China, September 21-23, 2018, Proceedings, Part I*, 2018. doi: 10.1007/978-981-13-2203-7_52.

[35] Y-H. Chen and J. Chen. Aintiphish — machine learning mechanisms for cyber-phishing attack. *IEICE Transactions on Information and Systems*, E102.D:878–887, 2019. doi: 10.1587/transinf.2018NTI0001.

[36] Kevin Clark, Minh-Thang Luong, and Quoc V Le. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*, 2020.

[37] British Columbia. Ten privacy principles. `https://www2.gov.bc.ca/gov/content/governments/services-for-government/information-management-technology/privacy/training/principles`. (accessed: 01.05.2023).

[38] Consultancy UK. Six privacy principles for general data protection regulation compliance. `https://www.consultancy.uk/news/13487/six-privacy-principles-for-general-data-protection-regulation-compliance`, June 1 2017.

[39] Cybertalents. What is Cybercrime? Types, Examples, and Prevention. `https://cybertalents.com/blog/what-is-cyber-crime-types-examples-and-prevention`. (accessed: 01.05.2023).

[40] S. Dargan, M. Kumar, M.R. Ayyagari, et al. A survey of deep learning and its applications: A new paradigm to machine learning. *Arch Computat Methods Eng*, 27:1071–1092, 2020. doi: 10.1007/s11831-019-09344-w.

[41] DataScientest. Long short term memory (lstm): de quoi s'agit-il ? https://datascientest.com/long-short-term-memory-tout-savoir. Last updated: October 2023.

[42] Jessica Davis. Covid-19 impact on ransomware, threats, healthcare cybersecurity, 2020. URL https://healthitsecurity.com/news/covid-19-impact-on-ransomware-threats-healthcare-cybersecurity. Accessed on March 2024.

[43] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[44] Nguyet Quang Do, Ali Selamat, Ondrej Krejcar, Enrique Herrera-Viedma, and Hamido Fujita. Deep learning for phishing detection: Taxonomy, current challenges and future directions. *IEEE Access*, 10:36429–36463, 2022. doi: 10.1109/ACCESS.2022.3151903.

[45] QN Do, A Selamat, O Krejcar, T Yokoi, and H Fujita. Phishing webpage classification via deep learning-based algorithms: An empirical study. *Applied Sciences*, 11(9210), 2021.

[46] M. Dunlop, S. Groat, and D. Shelly. Goldphish: using images for content-based phishing analysis. In *The Fifth International Conference on Internet Monitoring and Protection*, pages 123–128, Spain, 2010.

[47] M. Ebenezer and A. Arya. An atypical metaheuristic approach to recognize an optimal architecture of a neural network. In *Proceedings of the 14th International Conference on Agents and Artificial Intelligence (ICAART 2022) - Volume 3*, pages 917–925, 2022. ISBN 978-989-758-547-0. doi: 10.5220/0010951600003116.

[48] ESM El-Alfy. Detection of phishing websites based on probabilistic neural networks and k-medoids clustering. *The Computer Journal*, 60(12):1745–1759, 2017. doi: 10.109/comjn/bxx035. URL https://doi.org/10.109/comjn/bxx035.

[49] M. Elsadig, A.O. Ibrahim, S. Basheer, M.A. Alohali, S. Alshunaifi, H. Alqahtani, N. Alharbi, and W. Nagmeldin. Intelligent deep machine learning cyber phishing url detection based on bert features extraction. *Electronics*, 11:3647, 2022. doi: 10.3390/electronics11223647. URL https://doi.org/10.3390/electronics11223647.

[50] M. D. Faisal Khan and B. L. Rana. Detection of phishing websites using deep learning techniques. *Turkish Journal of Computer and Mathematics Education*, 12(10):3880–3892, 2021.

[51] Isabel Feher-Watters. What Tools Do Business Analysis Professionals Use to Manage Cybersecurity Analysis? https://www.iiba.org/business-analysis-blogs/what-tools-do-business-analysis-professionals-use-to-manage-cybersecurity-analysis/.

[52] Keith D. Foote. A brief history of deep learning. https://www.dataversity.net/brief-history-deep-learning/, Feb 2022. Dataversity.net.

[53] forcepoint.com. What is a Firewall? Firewalls defined, explained, and explored. https://www.forcepoint.com/cyber-edu/firewall. (accessed: 01.05.2023).

[54] fortinet.com. 19 Types of Phishing Attacks . https://www.fortinet.com/resources/cyberglossary/types-of-phishing-attacks. (accessed: 01.05.2023).

[55] World Economic Forum. Covid-19 risks outlook: A preliminary mapping and its implications, 2020. URL https://www.weforum.org/reports/covid-19-risks-outlook-a-preliminary-mapping-and-itsimplications. Accessed on March 2024.

[56] Sean Gallagher and Alex Brandt. Facing down the myriad threats tied to covid-19, 2020. URL https://news.sophos.com/en-us/2020/04/14/covidmalware. Accessed on March 2024.

[57] B. Guo, Y. Zhang, C. Xu, F. Shi, Y. Li, and M. Zhang. Hinphish: An effective phishing detection approach based on heterogeneous information networks. *Applied Sciences*, 11 (20):1–19, 2021.

[58] B.B. Gupta, N.A.G. Arachchilage, and K.E. Psannis. Defending against phishing attacks: taxonomy of methods, current issues, and future directions. *Telecommunication Systems*, 67:247–267, 2018. doi: 10.1007/s11235-017-0334-z. URL https:/doi.org/10.1007/s11235-017-0334-z.

[59] A. Hannousse and S. Yahiouche. Towards benchmark datasets for machine learning based website phishing detection: An experimental study. *Engineering Applications of Artificial Intelligence*, 104C:104347, 2021.

[60] Abdelhakim Hannousse and Salima Yahiouche. Web page phishing detection. https://data.mendeley.com/datasets/c2gw7fy2j4/3, 2021.

[61] Mahmudul Hasan, Md Milon Islam, Md Ishrak Islam Zarif, and MMA Hashem. Attack and anomaly detection in iot sensors in iot sites using machine learning approaches. *Internet of Things*, 7:100059, 2019. doi: 10.1016/j.iot.2019.100059.

[62] Tobias Hey, Jan Keim, Anne Koziolek, and Walter F. Tichy. NoRBERT: Transfer learning for requirements classification. In *2020 IEEE 28th International Requirements Engineering Conference (RE)*, pages 169–179. IEEE, 2020. doi: 10.1109/RE48521.2020.00028.

[63] Dau Hoang, Dang Minh, and Ninh Trang. A cnn-based model for detecting malicious urls. In *2023 RIVF International Conference on Computing and Communication Technologies At: Hanoi, Vietnam*, 12 2023.

[64] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, 1997.

[65] Raphael Hoheisel, Guido van Capelleveen, Dipti K Sarmah, and Marianne Junger. The development of phishing during the covid-19 pandemic: An analysis of over 1100 targeted domains. *Computers & Security*, 128:103158, 2023. doi: 10.1016/j.cose.2023.103158.

[66] hooksecurity. Phishing email examples. URL `https://www.hooksecurity.co/phishing-email-examples`. Accessed january 2024.

[67] Y. Huang, Q. Yang, J. Qin, and W. Wen. Phishing url detection via cnn and attention-based hierarchical rnn. In *Proceedings of the 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE, 2019.

[68] impicode.com. Artificial intelligence. URL `https://impicode.com/artificial-intelligence/`. Accessed january 2024.

[69] Md Sajadul Islam, Mst Jyoti, Md Mia, and Md Gulzar Hussain. Fake website detection using machine learning algorithms. 06 2023. doi: 10.1109/ICDATE58146.2023.10248584.

[70] itgovernance. What is Cyber Security? Definition and Best Practices. `https://www.itgovernance.co.uk/what-is-cybersecurity`. (accessed: 01.05.2023).

[71] Neelesh Karthikeyan. Step-by-step guide for a deep learning project. `https://medium.com/@neelesh_k/structuring-deep-learning-projects-b83d29513aea`. Last updated: February 2022.

[72] kaspersky.com. What is Cyber Security? . `https://www.kaspersky.com/resource-center/definitions/what-is-cyber-security`. (accessed: 01.05.2023).

[73] Kamaljit Kaur and Parminder Kaur. Bert-cnn: Improving bert for requirements classification using cnn. *Procedia Computer Science*, 218:2604–2611, 2023. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2023.01.234. URL https://www.sciencedirect.com/science/article/pii/S187705092300234X. International Conference on Machine Learning and Data Engineering.

[74] keras.io. Keras documentation: Distilbert tokenizer. https://keras.io/api/keras_nlp/models/distil_bert/distil_bert_tokenizer/. Accessed on: 2024-04-05.

[75] M. Khonji, Y. Iraqi, and A. Jones. Phishing detection: A literature survey. *IEEE Communications Surveys & Tutorials*, 15(4):2091–2121, 2013.

[76] L. Klusaite. What are scam websites, and how can you tell if a page is fake? https://nordvpn.com/blog/fake-scam-websites/.

[77] Nandhini Kumaran and Shilpa Lugani. Protecting businesses against cyber threats during covid-19 and beyond, 2020. URL https://cloud.google.com/blog/products/identity-security/protecting-against-cyber-threats-during-covid-19-and-beyond. Accessed on March 2024.

[78] A. Lakshmanarao, M. R. Babu, and M. M. Bala Krishna. Malicious url detection using nlp, machine learning and flask. In *2021 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES)*, pages 1–4, Chennai, India, 2021. doi: 10.1109/ICSES52305.2021.9633889.

[79] Harjinder Singh Lallie, Laura A Shepherd, Jason RC Nurse, Arnau Erola, Gregory Epiphaniou, Carsten Maple, and Xavier Bellekens. Cyber security in the age of covid-19: A timeline and analysis of cyber-crime and cyber-attacks during the pandemic. *Comput Secur*, 105: 102248, Jun 2021. doi: 10.1016/j.cose.2021.102248.

[80] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2020.

[81] Aleksandar Lazarevic, Vipin Kumar, and Jaideep Srivastava. *Intrusion Detection: A Survey*, volume 5, pages 19–78. 01 2005. ISBN 0-387-24226-0. doi: 10.1007/0-387-24230-9_2.

[82] H. Le, Q. Pham, D. Sahoo, and S. C. H. Hoi. Urlnet: Learning a url representation with deep learning for malicious url detection. In *Proceedings of ACM Conference 2017*. ACM, 2018.

[83] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521:436–444, 2015. doi: 10.1038/nature14539. URL https://doi.org/10.1038/nature14539.

[84] Yann LeCun, Y. Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–444, 2015. doi: 10.1038/nature14539.

[85] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[86] Richard Lush. Helping defend against a 30,000% increase in phishing attacks related to covid-19 scams, 2020. URL https://www.cgi-group.co.uk/en-gb/blog/cyber-security/helping-defend-against-a-30000-increase-in-phishing-attacks-related-to-covid-19-scams. Accessed on March 2024.

[87] Pierre-Louis Lussan. Les bases de la sécurité et de la protection des données. https://blog.netwrix.fr/2020/04/13/les-bases-de-la-securite-et-de-la-protection-des-donnees/. Last updated: 17 octobre 2022.

[88] Benjamin McConnell, Daniel Monaco, Mahdieh Zabihimayvan, Fatemeh Abdollahzadeh, and Samir Hamada. *Phishing Attack Detection: An Improved Performance Through Ensemble Learning*, pages 145–157. 09 2023. ISBN 978-3-031-42507-3. doi: 10.1007/978-3-031-42508-0_14.

[89] Samuel Rodríguez Medina. *Multi-Label Text Classification with Transfer Learning for Policy Documents: The Case of the Sustainable Development Goals*. Dissertation, 2019. URL https://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-395186.

[90] Natarajan Meghanathan. A tutorial on network security: Attacks and controls. 1, 12 2014. doi: 10.4018/978-1-4666-4789-3.ch011.

[91] S. Mohammadi and M. Babagoli. A novel hybrid hunger games algorithm for intrusion detection systems based on nonlinear regression modeling. *Int. J. Information Security*, 22: 1177–1195, 2023. doi: 10.1007/s10207-023-00684-0.

[92] Pascal Monasse and Kimia Nadjahi. Classez et segmentez des données visuelles. découvrez les différentes couches d'un cnn. https://openclassrooms.com/fr/courses/4470531-classez-et-segmentez-des-donnees-visuelles. Last updated: July 2022.

[93] Cedric Nabe. Impact of COVID-19 on Cybersecurity. https://www2.deloitte.com/ch/en/pages/risk/articles/impact-covid-cybersecurity.html. (accessed: 01.05.2023).

[94] K. Naeemullah, K. Ismael, and D. Elika. Improved feature selection method for features reduction in intrusion detection systems. *Mesop. J. CyberSecur.*, page 9–15, 2021.

[95] Ontario. Chapitre 7: Principes fondamentaux de la protection de la vie privée. https://www.ontario.ca/fr/document/manuel-sur-lacces-linformation-et-la-protection-de-la-vie-privee/chapitre-7-principes-fondamentaux-de-la-protection-de-la-vie-privee#. (accessed: 01.05.2023).

[96] C. Opara, Bo. Wei, and Y. Chen. Htmlphish: Enabling phishing web page detection by applying deep learning techniques on html analysis. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020.

[97] Mohan Rao P, S. Murali Krishna, and Siva Kumar. A. *Modern Privacy Threats and Privacy Preservatio Techniques in Data Analytics*, chapter 4. Factoring Ethics in Technology, Policy Making, Regulation and AI. IntechOpen, 2021.

[98] Santi P. and Dr. B. Indira R. Phishing and anti phishing techniques. *International Research Journal of Engineering and Technology (IRJET)*, 06(07), 2019. ISSN 2395-0072.

[99] DLA PIPER. Data Protection Laws of the World-National Data Protection Authority. https://www.dlapiperdataprotection.com/index.html?t=authority&c=DZ.

[100] pro.arcgis.com. Introduction to deep learning. URL https://pro.arcgis.com/en/pro-app/latest/help/analysis/deep-learning/what-is-deep-learning-.htm. Accessed january 2024.

[101] A. G. Putrada, N. Alamsyah, S. F. Pane, and M. N. Fauzan. XGBoost for IDS on WSN Cyber Attacks with Imbalanced Data. In *2022 International Symposium on Electronics and Smart Devices (ISESD)*, pages 1–7, Bandung, Indonesia, 2022. doi: 10.1109/ISESD56103.2022.9980630.

[102] Python Basics. What is flask? https://pythonbasics.org/what-is-flask-python/. Accessed February 2024.

[103] Alfio Quarteroni. *A Bit of Maths (Behind Artificial Intelligence and Machine Learning)*, pages 43–53. 01 2022. ISBN 978-3-030-96165-7. doi: 10.1007/978-3-030-96166-4_5.

[104] Prachit R, Harshal V, and Rishikesh S. A survey of phishing website detection systems. *International Research Journal of Engineering and Technology (IRJET)*, 07(11), 2020.

[105] Zaimi R, M. Hafidi, and L Mahnane. Survey paper: Taxonomy of website anti-phishing solutions. In *Seventh International Conference on Social Networks Analysis, Management and Security (SNAMS2020). IEEE France*, pages 1–8, 2020. doi: 10.1109/SNAMS52053.2020.9336559.

[106] Zaimi R, M. Hafidi, and L Mahnane. A literature survey on anti-phishing in websites. In *The 4th International Conference on Networking, Information Systems & Security (NISS2021). ACM*, 2021. doi: 10.1145/3454127.3456580. URL `https://doi.org/10.1145/3454127.3456580`.

[107] Zaimi R, M. Hafidi, and L Mahnane. A deep learning approach to detect phishing websites using cnn for privacy protection. *Intelligent Decision Technologies*, 17(3):713–728, 2023. doi: 10.3233/IDT-220307.

[108] Zaimi R, M. Hafidi, and L Mahnane. A deep learning mechanism to detect phishing urls using the permutation importance method and smote-tomek link. *J Supercomput. Springer*, 80, 12, 2024. doi: https://doi.org/10.1007/s11227-024-06124-7.

[109] Zaimi R, Safi Eljil. K, M. Hafidi, L Mahnane, and Nait abdesselam F. An enhanced mechanism for malicious url detection using deep learning and distilbert-based feature extraction. *J Supercomput. Springer*, 81, 438, 2025. doi: https://doi.org/10.1007/s11227-024-06908-x.

[110] Mijanur Rahman. Different ways to combine cnn and lstm networks for time series classification tasks. Medium.com, 2022. URL `https://medium.com/mixany/differentwaysto-combine-cnn-and-lstm-networks-for-time-seriesclassification-tasksb03fc`

[111] Vishal Rajput. Pooling layers in neural nets and their variants. `https://medium.com/aiguys/pooling-layers-in-neural-nets-and-their-variants-f6129fc4628b`. Last updated: Jan 10, 2022.

[112] S. H. Rashid and W. D. Abdullah. Cloud-based machine learning approach for accurate detection of website phishing. *Int. J. Intell. Syst. Appl. Eng.*, 11:451–460, 2023.

[113] G. Nikhita Reddy and G. J. Ugander Reddy. A study of cyber security challenges and its emerging trends on latest technologies. *ArXiv*, abs/1402.1842, 2014. URL `https://api.semanticscholar.org/CorpusID:6758838`.

[114] ND Rohith. Structure of url. `https://dev.to/ndrohith/structure-of-url-2n9c`. Posted on Apr 8, 2022.

[115] A. Safi and S. Singh. A systematic literature review on phishing website detection techniques. *Journal of King Saud University-Computer and Information Sciences*, 35(2):590–611, 2023.

[116] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[117] Frank Shi. Threat spotlight: Coronavirus-related phishing, 2020. URL `https://blog.barracuda.com/2020/03/26/threat-spotlight-coronavirus-related-phishing`. Accessed on March 2024.

[118] Manu Siddhartha. Malicious urls dataset. `https://www.kaggle.com/datasets/sid321axn/malicious-urls-dataset`. Accessed on Sept 2023.

[119] Milita Songailaitė, Eglė Kankevičiūtė, Bohdan Zhyhun, and Justina Mandravickaitė. Bert-based models for phishing detection. In *28th Conference on Information Society and University Studies (IVUS'2023)*, Kaunas, Lithuania, May 2023. CEUR Workshop Proceedings (CEUR-WS.org).

[120] G. Sonowal and K.S. Kuppusamy. Phidma - a phishing detection model with multi-filter approach. *Journal of King Saud University - Computer and Information Sciences*, 32(1): 99–112, 2017. doi: 10.1016/j.jksuci.2017.07.005.

[121] Bc. Radek Starosta. Phishing detection using natural language processing. Master's thesis, Czech Technical University in Prague, 2021.

[122] Winston & Strawn. What is online privacy? `https://www.winston.com/en/legal-glossary/online-privacy`, 2023. (accessed: 01.05.2023).

[123] J.R. Tietsort and A Benny. 17 Types of Cyber Attacks Commonly Used By Hackers. `https://www.aura.com/learn/types-of-cyber-attacks`.

[124] T. Tiwari, T. Tiwari, and S. Tiwari. How artificial intelligence, machine learning and deep learning are radically different? *International Journal of Advanced Research in Computer Science and Software Engineering*, 8(2):1–9, 2018.

[125] Tarun Tiwari. Phishing site urls dataset. `https://www.kaggle.com/datasets/taruntiwarihp/phishing-site-urls`. Accessed on Sept 2023.

[126] B. Tomasz, C-E. Valentín, S-P. Josep, and Pere B-R. A survey on web tracking: Mechanisms, implications, and defenses. *Proceedings of the IEEE*, 105(8):1476–1510, 2017.

[127] M. M. Uddin, K. Arfatul Islam, M. Mamun, V. K. Tiwari, and J. Park. A comparative analysis of machine learning-based website phishing detection using url information. In *2022 5th International Conference on Pattern Recognition and Artificial Intelligence (PRAI)*, pages 220–224, Chengdu, China, 2022. doi: 10.1109/PRAI55851.2022.9904055.

[128] unctad. Cybercrime Legislation Worldwide. `https://unctad.org/page/cybercrime-legislation-worldwide`. (accessed: 01.05.2023).

[129] D. M. Vargheese and N. R. Sreelakshmi. Phishing website detection using machine learning techniques and cnn. *International Journal of Engineering Research & Technology*, 10(4): 120–123, 2022.

[130] G. Varshney, M. Misra, and P. Atrey. A survey and classification of web phishing detection schemes. *Journal of Security and Communication Networks*, 9:6266–6284, 2016.

[131] Raden Aurelius Andhika Viadinugroho. Imbalanced classification in python: Smote-tomek links method. `https://towardsdatascience.com/imbalanced-classification-in-python-smote-tomek-links-method-6e48dfe69bbc`. Posted on Apr 18, 2021.

[132] vib.be. Six principles from the european general data protection regulation. `https://elearning.vib.be/courses/writing-a-data-management-plan/lessons/privacy/topic/six-principles-from-the-european-general-data-protection-regulation/`. Accessed on: December 2023.

[133] Grega Vrbančič. Phishing websites dataset. `https://data.mendeley.com/datasets/72ptz43s9v/1`, 2020.

[134] Y. Wei and Y. Sekiya. Feature selection approach for phishing detection based on machine learning. In *Proceedings of the International Conference on Applied CyberSecurity (ACS) 2021*, pages 61–70, 2022. doi: 10.1007/97830309591807.

[135] Y. Wei and Y. Sekiya. Feature selection approach for phishing detection based on machine learning. In *Proceedings of the International Conference on Applied CyberSecurity (ACS) 2021*, pages 61–70, 2022. doi: 10.1007/9783030-9591807.

[136] X. Xiao, W. Xiao, D. Zhang, B. Zhang, et al. Phishing websites detection via cnn and multi-head self-attention on imbalanced datasets. *Computers & Security*, 108:0167–4048, 2021.

[137] Sarcastic Writer/ yeahhub.com. About Wireshark – A Packet Sniffer and its Components. https://www.yeahhub.com/about-wireshark-a-packet-sniffer-and-its-components/.

[138] Y. Zhang, J. Hong, and L. Cranor. Cantina: A content-based approach to detecting phishing web sites. In *16th International Conference on World Wide Web*, pages 639–648, 2007.

[139] J. Zhu and J. Song. An intelligent classification model for surface defects on cement concrete bridges. *Applied Sciences*, 10(3):972, 2020. doi: 10.3390/app10030972.

[140] P. Zimmermann. Web of trust (wot), 1992. URL https://www.mywot.com/en/aboutus. Accessed October 2023.

Chapter. A

Appendix 1: Features list of the "Phishing Websites Dataset from Mendeley Data"

Table A.1: Features symbols and names of the "Phishing Websites Dataset from Mendeley Data" [133] 2.

Symbol	Feature
F1	qty_dot_url
F2	qty_hyphen_url
F3	qty_underline_url
F4	qty_slash_url
F5	qty_questionmark_url
F6	qty_equal_url
F7	qty_at_url
F8	qty_and_url
F9	qty_exclamation_url
F10	qty_space_url
F11	qty_tilde_url
F12	qty_comma_url
F13	qty_plus_url
F14	qty_asterisk_url
F15	qty_hashtag_url
F16	qty_dollar_url
F17	qty_percent_url
F18	qty_tld_url
F19	length_url

Continued on next page

Table A.1 – *Continued from previous page*

Symbol	Feature
F20	qty_dot_domain
F21	qty_hyphen_domain
F22	qty_underline_domain
F23	qty_slash_domain
F24	qty_questionmark_domain
F25	qty_equal_domain
F26	qty_at_domain
F27	qty_and_domain
F28	qty_exclamation_domain
F29	qty_space_domain
F30	qty_tilde_domain
F31	qty_comma_domain
F32	qty_plus_domain
F33	qty_asterisk_domain
F34	qty_hashtag_domain
F35	qty_dollar_domain
F36	qty_percent_domain
F37	qty_vowels_domain
F38	domain_length
F39	domain_in_ip
F40	server_client_domain
F41	qty_dot_directory
F42	qty_hyphen_directory
F43	qty_underline_directory
F44	qty_slash_directory
F45	qty_questionmark_directory
F46	qty_equal_directory
F47	qty_at_directory
F48	qty_and_directory
F49	qty_exclamation_directory
F50	qty_space_directory
F51	qty_tilde_directory
F52	qty_comma_directory
F53	qty_plus_directory
F54	qty_asterisk_directory
F55	qty_hashtag_directory
F56	qty_dollar_directory
F57	qty_percent_directory

Continued on next page

Table A.1 – *Continued from previous page*

Symbol	Feature
F58	directory_length
F59	qty_dot_file
F60	qty_hyphen_file
F61	qty_underline_file
F62	qty_slash_file
F63	qty_questionmark_file
F64	qty_equal_file
F65	qty_at_file
F66	qty_and_file
F67	qty_exclamation_file
F68	qty_space_file
F69	qty_tilde_file
F70	qty_comma_file
F71	qty_plus_file
F72	qty_asterisk_file
F73	qty_hashtag_file
F74	qty_dollar_file
F75	qty_percent_file
F76	file_length
F77	qty_dot_params
F78	qty_hyphen_params
F79	qty_underline_params
F80	qty_slash_params
F81	qty_questionmark_params
F82	qty_equal_params
F83	qty_at_params
F84	qty_and_params
F85	qty_exclamation_params
F86	qty_space_params
F87	qty_tilde_params
F88	qty_comma_params
F89	qty_plus_params
F90	qty_asterisk_params
F91	qty_hashtag_params
F92	qty_dollar_params
F93	qty_percent_params
F94	params_length
F95	tld_present_params

Continued on next page

Table A.1 – *Continued from previous page*

Symbol	**Feature**
F96	qty_params
F97	email_in_url
F98	time_response
F99	domain_spf
F100	asn_ip
F101	time_domain_activation
F102	time_domain_expiration
F103	qty_ip_resolved
F104	qty_nameservers
F105	qty_mx_servers
F106	ttl_hostname
F107	tls_ssl_certificate
F108	qty_redirects
F109	url_google_index
F110	domain_google_index
F111	url_shortened

Chapter. B

Appendix 2: List of Scientific Productions

B.1 International Conference Papers

- Zaimi. R, Hafidi. M, and Mahnane. L. "Survey paper: Taxonomy of website anti-phishing solutions". In the Seventh International Conference on Social Networks Analysis, Management and Security (SNAMS2020).IEEE, pages 1–8, 2020. doi: 10.1109/SNAMS52053.2020.9336559. Paris, France.

- Zaimi. R, Hafidi. M, and Mahnane. L. "A literature survey on anti-phishing in websites". In The 4th International Conference on Networking, Information Systems & Security (NISS2021), 2021. ACM. doi: 10.1145/3454127.3456580. URL https://doi.org/10.1145/3454127.3456580. Kenitra. Morocco.

- Zaimi. R, Hafidi. M, and Mahnane. L. "Phishing website detection using deep learning and machine learning". ICASA'22 «First International Conference on Autonomous Systems and their Applications» 2022. Chadli Bendjedid el Tarf University. Algeria.

B.2 Journal Articles

- Zaimi. R, Hafidi. M, and Mahnane. L. «A deep learning approach to detect phishing websites using cnn for privacy protection». Intelligent Decision Technologies, 17(3):713–728, (2023). doi: 10.3233/IDT-220307.

- Zaimi. R, Hafidi. M, and Mahnane. L. «A deep learning mechanism to detect phishing URLs using the permutation importance method and SMOTE-Tomek link». J Supercomput. Springer 80, 12(2024). https://doi.org/10.1007/s11227-024-06124-7

• Zaimi. R, Safi Eljil. K, Hafidi. M, Mahnane. L and, Nait abdesselam. F. «An enhanced mechanism for malicious URL detection using deep learning and DistilBERT-based feature extraction». J Supercomput 81, 438 (2025). https://doi.org/10.1007/s11227-024-06908-x