DETECTION OF PHISHING WEB PAGES BY COMBINING SEMANTICAL AND VISUAL INFORMATION

KİMLİK AVCISI WEB SAYFALARININ ANLAMSAL VE GÖRSEL BİLGİYLE TESPİTİ

AHMAD HANI ABDALLA ALMAKHAMREH

ASST. PROF. DR. AHMET SELMAN BOZKIR
Supervisor

Submitted to
Graduate School of Science and Engineering of Hacettepe University
as a Partial Fulfillment to the Requirements
for the Award of the Degree of Master of Science
in Computer Engineering

April 2024

ABSTRACT

DETECTION OF PHISHING WEB PAGES BY COMBINING SEMANTICAL AND VISUAL INFORMATION

Ahmad Hani Abdalla Almakhamreh
Master of Science, Computer Engineering
Supervisor: Asst. Prof. Dr. Ahmet Selman BOZKIR
April 2024, 134 pages

The increased frequency and sophistication of cybercrimes resulted in severe monetary loss for individuals and entities, increasing the demand for robust and sustainable solutions. Although there are countless anti-phishing solutions in the domain, cybercriminals exploit these systems and bypass them with zero-day attacks. In this dissertation, a new end-to-end deep learning model called *CrossPhire* is proposed, which uses semantic and visual features to make machine learning-based classification between phishing and legitimate web pages. *CrossPhire* extracts distinctive features from three different data environments, including URLs, source code, and screenshots obtained from web pages, and is jointly trained. In this work, we present the following novelties: (1) development of an end-to-end deep learning model capable of capturing semantic and visual features from the page's URL, plain textual content, and screenshot, (2) a language-independent analysis approach, leveraging SOTA sentence transformers and convolutional neural networks, enabling analysis without reliance on third-party services, (3) a new highly diverse multimedia dataset compiling real-world examples of legitimate and phishing web pages, called *Phish360*, (4) provision of statistical reports based on extensive data analysis[1] of Phish360 and other multimodal datasets in the

[1]**Data Analysis Reports**: https://github.com/almakhamreh/MM-datasets-EDA

literature, and (5) conducting comprehensive experiments, including in-data and cross-data validation across five different datasets to evaluate the generalization performance of the proposed model.

A comprehensive series of experiments was conducted to identify the most effective model configuration by exploring various combinations of (a) HTML parsers (BeautifulSoup and Trafilatura), (b) sentence transformers (Sentence-BERT and multilingual XLM-R), and (c) convolutional image classifier models (ResNet50 and DenseNet121). In the experiments, *CrossPhire* demonstrated outstanding performance, achieving 99.21% accuracy on the Phish360 dataset and maintaining an average accuracy of 99.26% across the four benchmark datasets. Additionally, we fine-tuned the CLIP model using the available benchmark datasets by integrating a two-hidden layer MLP. Our approach demonstrated superior results compared to CLIP, consistently outperforming it across all employed datasets. These findings establish *CrossPhire* as a highly effective solution across various scales and datasets, surpassing existing approaches.

Keywords: cybersecurity, phishing detection, multimodality, natural language processing, computer vision, transfer learning, deep learning, zero-day attacks.

ÖZET

KİMLİK AVCISI WEB SAYFALARININ ANLAMSAL VE GÖRSEL BİLGİYLE TESPİTİ

Ahmad Hani Abdalla Almakhamreh
Yüksek Lisans, Bilgisayar Mühendisliği
Danışman: Asst. Prof. Dr. Ahmet Selman BOZKIR
Nisan 2024, 134 sayfa

Siber suçların artan sıklığı ve karmaşıklığı bireyler ve kurumlar için güvenlik zaafiyetleri ile birlikte ciddi maddi kayıplarla sonuçlanmakta ve bunun sonucunda gürbüz ve sürdürülebilir çözümlere olan talebi artırmıştır. Alanda sayısız kimlik avı önleme çözümü olmasına rağmen, saldırganlar bu sistemleri istismar etmekte ve sıfırıncı gün saldırılarıyla bunları atlatmaktadır. Bu tez çalışmasında, kimlik avcısı ve meşru web sayfaları arasında makine öğrenimine dayalı sınıflandırma yapmak için anlamsal ve görsel özellikleri kullanan *CrossPhire* isimli yeni bir uçtan uca derin öğrenme modeli önerilmiştir. *CrossPhire*, web sayfalarından elde edilen URL, kaynak kod ve ekran görüntüleri olmak üzere üç farklı ortamdan ayırt edici özellikler çıkarmakta ve bütünleşik bir öğrenme yöntemiyle eğitilmektedir. Bu çalışmada maddeler halinde şu katkılar sunulmuştur: (1) Sayfanın URL ve temel metinsel içeriği ile web sayfası şipşakından anlamsal ve görsel özellikleri yakalayan uçtan uca derin öğrenme modeli, (2) üçüncü taraf hizmetlerden izole olarak, güncel "cümle dönüştürücüler" ve evrişimsel sinir ağları yardımıyla dilden bağımsız bir analiz yöntemi, (3) *Phish360* adı verilmiş olan meşru ve oltalayıcı sayfaların yer aldığı gerçek dünya örneklerinin derlendiği, çeşitliliği yüksek yeni bir çok ortamlı veri kümesi, (4) *Phish360* ve literatürde yer alan diğer veri kümelerinin veri analizine dayalı istatistiksel raporlar[2] ve (5) önerilen modelin genelleme başarımını ölçmek adına beş farklı veri kümesiyle iç-veri ve çapraz-veri doğrulamasına dayalı kapsamlı deneyler.

En iyi modelin bulunması adına farklı (a) HTML ayrıştırıcılar (BeautifulSoup ve Trafilatura), (b) cümle dönüştürücüleri (Sentence-BERT ve çok dilli XLM-R) ve (c) imge sınıflayıcı evrişşimsel modeller (ResNet50 ve DenseNet121) arasındaki kombinasyonlar kapsamlı deneylerle ölçümlenmiştir. Yapılan deneylerde *CrossPhire*, Phish360 veri kümesinde 99,21% doğruluk sunarken ve diğer dört kıyaslama veri kümesinde ortalama 99,26% doğruluk başarımı elde edilmiştir. Ek olarak, iki gizli katmanlı MLP'yi entegre ederek mevcut kıyaslama veri kümelerini kullanarak CLIP modelinde ince ayar yaptık. Yaklaşımımız, CLIP'e kıyasla üstün sonuçlar ortaya koydu ve kullanılan tüm veri kümelerinde sürekli olarak CLIP'ten daha iyi performans göstermiştir. Sonuç olarak *CrossPhire*'ın, kullanılan tüm veri kümelerinde farklı ölçeklerin tamamında en yüksek sonuçları yakaladığı saptanmıştır.

Keywords: siber güvenlik, kimlik avı tespiti, çok ortamlılık doğal dil işleme, bilgisayar görüsü, transfer öğrenimi, derin öğrenme, sıfırıcı gün saldırıları

[2]**Veri analiz raporları**: https://github.com/almakhamreh/MM-datasets-EDA

ACKNOWLEDGEMENTS

Alhamdulillah, I express my deepest gratitude to Allah for granting me guidance, strength, and perseverance throughout this journey. I am profoundly grateful to my loving family for their unwavering support, encouragement, and understanding during this journey. Their belief in me has been my motivation, and I owe them immeasurable gratitude for their sacrifices and constant encouragement.

I extend my heartfelt appreciation to my advisor, Asst. Prof. Dr. Ahmet Selman Bozkır, for his invaluable guidance, mentorship, and unwavering support throughout this research. His expertise, patience, and encouragement have been instrumental in shaping my research and academic growth. Your commitment to academic excellence and willingness to share your knowledge have shaped my understanding and passion for this subject.

I am equally thankful to Assoc. Prof. Dr. Murat Aydos for his invaluable contributions, guidance, and support. I am truly grateful for your mentorship, collaborative spirit, and insightful feedback.

I would also like to express my gratitude to Prof. Dr. Mehmet Tekerek and the members of the thesis defense jury. Your insightful feedback and constructive criticism have significantly contributed to the refinement of this work. I am grateful for the time and effort you dedicated to evaluating and enhancing the quality of my research.

I also extend my gratitude to my colleagues and friends who have supported me in various ways during this journey. Your encouragement and insights have been invaluable to me.

I am also grateful to my past supervisor and mentor, Asst. Prof. Dr. Ozkan Kilic, for his invaluable assistance and guidance during my bachelor's years. His support and wisdom have significantly shaped my academic and personal journey.

I dedicate this dissertation to my father, Hani Almakhamreh, my mother, Hayat Abhary, and my grandfather, Abdullah Almakhamreh.

CONTENTS

Page

TABLES

FIGURES

ABBREVIATIONS

DL	:	**D**eep **L**earning
RL	:	**R**einforcement **L**earning
ML	:	**M**achine **L**earning
NLP	:	**N**atural **L**anguage **P**rocessing
Faster-RCNN	:	**Faster R**egion-based **C**onvolutional **N**eural **N**etworks
URL	:	**D**eep **L**earning
HTML	:	**H**yper**T**ext **M**arkup **L**anguage
CSS	:	**C**ascading **S**tyle **S**heets
Char	:	**Char**acter
DOM	:	**D**ocument **O**bject **M**odel
BERT	:	**B**idirectional **E**ncoder **R**epresentations from **T**ransformers
ELECTRA	:	**E**fficiently **L**earning an **E**ncoder that **C**lassifies **T**oken **R**eplacements **A**ccurately
SBERT	:	**S**entence-**BERT**
XLM-R	:	**XLM-R**oBERTa
DenseNet	:	**Dense**ly Connected Convolutional **Net**works
ResNet	:	**Res**idual **Net**works
LSTM	:	**L**ong **S**hort-**T**erm **M**emory
Bi-LSTM	:	**Bi**directional **L**ong **S**hort-**T**erm **M**emory
CNN	:	**C**onvolutional **N**eural **N**etwork
ANN	:	**A**rtificial **N**eural **N**etwork
DNN	:	**D**eep **N**eural **N**etwork
RNN	:	**R**ecurrent **N**eural **N**etwork
BiGRU	:	**Bi**directional **G**ated **R**ecurrent **U**nit
MMOD	:	**M**ax-**M**argin **O**bject **D**etection
HOG	:	**H**istogram of **O**riented **G**radients
DCP	:	**D**ominant **C**olor **P**ercentage

SSIM	:	**S**tructural **S**imilarity **I**ndex **M**easure
AWL	:	**Ab**stract **W**ebpage **L**ayout
OCR	:	**O**ptical **C**haracter **R**ecognition
CRP	:	**C**redential **R**equiring **P**age
CAPTCHA	:	**C**ompletely **A**utomated **P**ublic **T**uring test to tell **C**omputers and **H**umans **A**part
UI	:	**U**ser **I**nterface
QR	:	**Q**uick **R**esponse
PSHCS	:	**P**hishing **S**ites **H**osted on **C**ompromised **S**ervers
CBAM	:	**C**onvolutional **B**lock **A**ttention **M**odule
MCDM	:	**M**ultiple **C**riteria **D**ecision **M**aking
CSVs	:	**C**omma-**S**eparated **V**alues
EDA	:	**E**xploratory **D**ata **A**nalysis
TLD	:	**T**op **L**evel **D**omain
ccTLD	:	**c**ountry **c**ode **T**op **L**evel **D**omain
FLD	:	**F**ree-**L**evel **D**omain
TF	:	**T**ra**f**ilatura
BS	:	**B**eautiful**S**oup
Acc.	:	**Acc**uracy
ST	:	**S**entence **T**ransformer
t-SNE	:	**t**-distributed **S**tochastic **N**eighbor **E**mbedding
TAGCN	:	**T**opology **A**daptive **G**raph **C**onvolutional **N**etwork
GCN	:	**G**raph **C**onvolutional **N**etworks
LR	:	**L**ogistic **R**egression
XGB	:	**XGB**oost
SVM	:	**S**upport **V**ector **M**achine
LightGBM	:	**Light** **G**radient-**B**oosting **M**achine
CatB	:	**CatB**oost
KNN	:	**K**-**N**earest **N**eighbors
FC layer	:	**F**ully **C**onnected layer

IoT	:	**I**nternet **o**f **T**hings
DoS	:	**D**enial **o**f **S**ervice
DNS	:	**D**omain **N**ame **S**ystem
SQL	:	**S**tructured **Q**uery **L**anguage
MitM	:	**M**an **i**n **t**he **M**iddle
IP	:	**I**nternet **P**rotocol
HTTPS	:	**H**ypertext **T**ransfer **P**rotocol **S**ecure
TF-IDF	:	**T**erm **F**requency-**I**nverse **D**ocument **F**requency
DMOZ	:	**D**irectory **Moz**illa
LLMs	:	**L**arge **L**anguage **M**odels

1. INTRODUCTION

In the past decade, rapid technological advancements have led to widespread digitalization. Adapting to the constant developments, businesses, institutions, and individuals have shifted towards the digital world. As a result, the number of internet users has increased by 133% between 2010 and 2020 [6]. Humans have become progressively dependent on internet-based services like social media, banking systems, online news organizations, educational institutions, and even some governmental services, as it eases their lives and saves time. This widespread use of internet-based solutions opened the door for attackers to capitalize on by threatening users and service providers with multiple cybercrimes and phishing is one of them.

Phishing is a severe cybercrime that leverages deceptive websites, emails, and text messages to mislead people and steal sensitive information like login credentials, email addresses, and credit card information for financial gain. According to a study about the definition of phishing in 2014 [7], they defined it as "*a scalable act of deception whereby impersonation is used to obtain information from a target*" (as cited in [8]). In other words, attackers try to deceive and lure targets by impersonating a well-known, trustworthy entity. In fact, the more famous a brand is, the more likely attackers will attempt to impersonate it. That is because there is a higher chance that the target is using this famous brand's service (e.g., Microsoft, AT&T, and PayPal), so it would be more convincing to the victim.

Although phishing is an old tactic, it is still popular because it is simple and effective. According to CISCO's Cyber security threat trends in 2021, phishing accounted for 90% of data breaches [9]. Attackers typically target the weakest link in the security chain, i.e., the user [9]. As a result, the average internet user suffers the most from cybersecurity crimes. That is because attackers tend to exploit human vulnerabilities instead of software vulnerabilities. This means that even if a system is technically secure enough, end users may be tricked into accidentally disclosing personal information, which undermines the system's overall security. There is no shortage of examples demonstrating the damage done

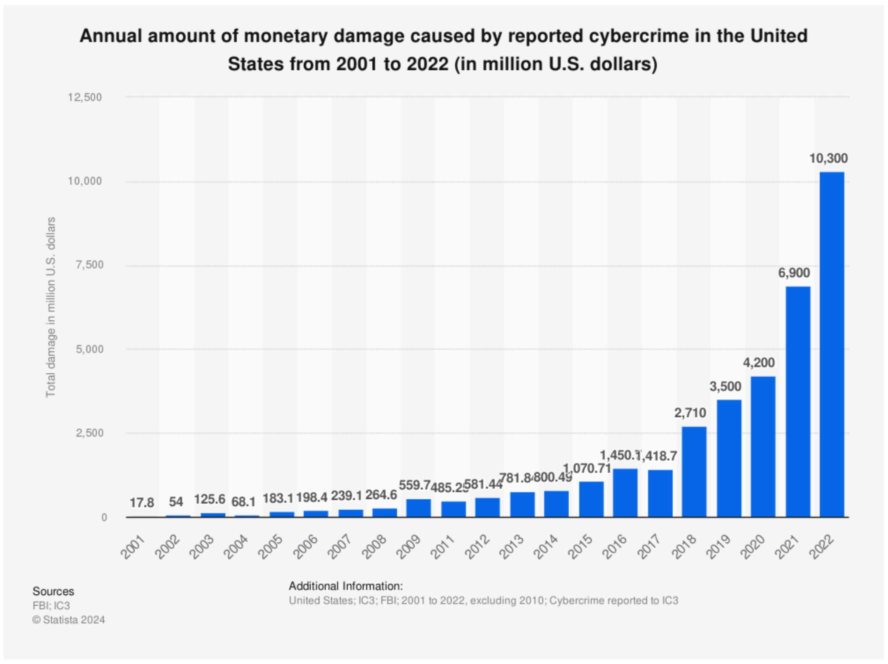


Figure 1.1 Annual financial loss caused by reported cybercrime in the United States from 2001 to 2022 [1].

by cybersecurity attacks. In the US in 2022 alone, more than 800 thousand complaints were made to the FBI's Internet Crime Complaint Center (IC3), resulting in a total potential loss of $10.2 billion [10]. Which is considerably greater than the total loss in 2021, estimated at $6.9 billion from ∼847 thousand complaints [11]. This shows that phishing attacks are becoming more sophisticated leading to a higher attack success rate. Meaning that the trend does not follow a direct correlation, but rather, an inverse correlation with more advanced tactics focused on the targeted victims. As demonstrated in Fig. 1.1, the annual loss from cybercrime in the United States is rapidly increasing, indicating that anti-phishing systems are not keeping up with the advancements of cyberattacks.

There are two primary techniques in phishing: *Deceptive phishing* and *Malware-based phishing*. Malware-based phishing aims to install malicious software or code using technical subterfuge schemes (expand this point)[12]. On the other hand, deceptive phishing employs techniques like social engineering to lure and deceive the targets into submitting their sensitive information [12]. In this thesis, we focus on deceptive phishing attacks and their countermeasure techniques, as they are employed more frequently, and the overall reach of

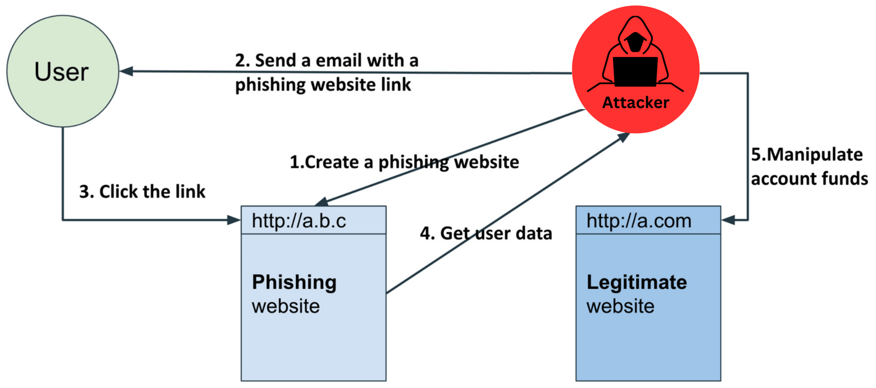


Figure 1.2 The lifecycle of a landing web page based phishing attack

these attacks is far greater.

In deceptive phishing, attackers create fraudulent websites having input fields to obtain the victim's sensitive information. Then, they execute the phishing campaigns by sending out spoofed emails or SMS in bulk, containing the malicious Uniform Resource Locator (URL) with catchy subject lines like 'Urgent', 'Notice', or 'Password change required'. It is similar to how a fisherman places bait in a lake with a high fish population, hoping to catch fish. Therefore, this type of phishing is commonly referred to as *Bulk Phishing*. As demonstrated in Fig. 1.2, the phishing attack cycle starts with a planning and website setup phase, followed by the distribution of fraudulent websites via email or SMS, typically chosen for its ease of reaching a large number of potential victims [13]. If the target visits the phishing website and submits his credentials, the attackers immediately use this sensitive information such as credit card details and login credentials for malicious purposes.

According to the Anti-phishing Working Group (APWG), the financial sector is one of the highly targeted sectors in the second Quarter of 2023 [14]. We analyzed the APWG reports from 2012 to 2023, showcasing the number of unique phishing websites created annually by attackers. As demonstrated in Fig. 1.3, there has been a dramatic surge in the number of phishing websites since 2020. Notably, between 2020 and 2023, there has been a staggering 228% increase, with the number of phishing websites reaching almost 5 Million. Given the substantial financial implications and potential losses associated with these types of

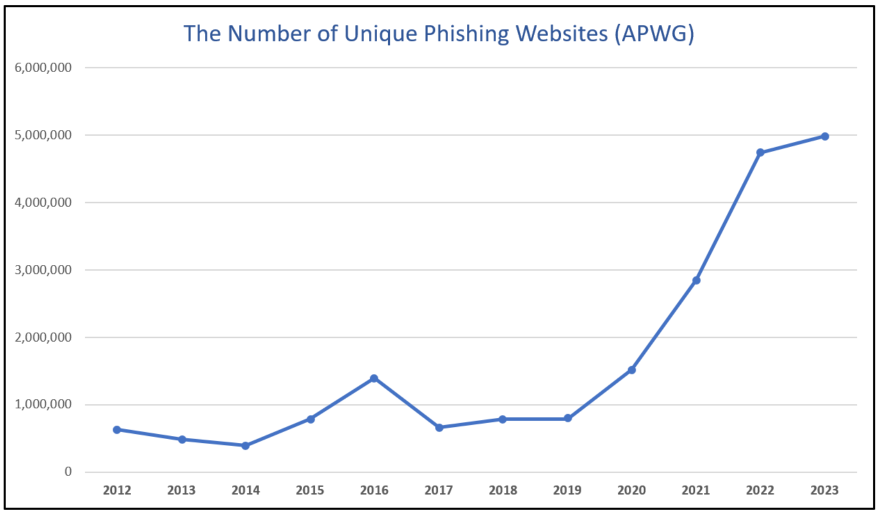


Figure 1.3 The number of unique phishing websites detected by APWG (2012 - 2023).

phishing attacks, there is a desperate need for robust anti-phishing mechanisms to protect both enterprises and individual users [15].

Numerous anti-phishing solutions in the literature address the different aspects of phishing detection. Given the phishing attacks' evolving and dynamic nature, researchers are constantly battling in an attempt to adapt to new emerging phishing tactics. Early works on phishing detection involved white and blacklisting (List-based) using URLs [16, 17]. However, the major disadvantage of these techniques is a minor change in the URL is sufficient to bypass it [16]. In addition, List-based approaches lack the learning ability to generalize on new URL samples. Later on, visual similarity approaches emerged to mitigate the phishing detection problem. Like Vision-based approaches [18–20] and approaches utilizing the structure of Source Code [21, 22]. More recently, machine learning-based (ML) solutions have also been experimented with, using handcrafted features [23, 24], TF-IDF [25] and N-gram features [26]. Nevertheless, handcrafted features demand extensive domain knowledge and continuous updates to keep up with new phishing tactics [27]. Additionally, the performance of models trained with handcrafted features has been shown to decrease when tested on new samples [26]. In general, single-modal approaches (using single data modality) are easier for attackers to bypass since the reliance on a single data source creates a vulnerability. For this reason, cybersecurity researchers are shifting towards hybrid approaches, combining data modalities to create a more robust anti-phishing solution.

Despite the high performance achieved by ML anti-phishing solutions, phishing campaigns are growing and can still bypass detection systems. Given the recent developments in deep learning, it is apparent that researchers are shifting towards utilizing deep learning approaches as they have been proven to outperform conventional ML algorithms in various domains. Additionally, scholars have contributed with large anti-phishing datasets in different categories like URL datasets [2, 25, 26, 28], image datasets [18, 29], and multimodal datasets [29–32]. This contribution paves the way for new deep learning-based proposals. The large anti-phishing datasets facilitate employing deep learning solutions with automatic feature extraction rather than using handcrafted features. These multimodal datasets contain the raw data modalities, unlike other anti-phishing datasets where the extracted features are shared. The availability of raw multimodal datasets provides freedom and flexibility to research scientists, where they can extract the relevant features to their study. Hence, encouraging scholars to propose and evaluate new methodologies without needing to collect their data, as it is a time-consuming and tedious job.

In this thesis, we conduct a comprehensive literature review while focusing on the recent novel anti-phishing approaches. Although there are several different categorizations of anti-phishing works in the literature. However, we categorize the explored anti-phishing solutions into five major categories based on the data modality employed:

1. URL-based approaches.
2. Content-based approaches.
3. Image-based approaches.
4. Bimodal approaches (i.e., use of two different modalities).
5. Multimodal approaches (i.e., Use of more than two modalities).

Conducting a thorough literature review helped us identify the following gaps within the anti-phishing research domain:
(i) A notable scarcity of multimodal anti-phishing benchmark datasets.
(ii) challenges associated with adapting anti-phishing systems to evolving phishing tactics,

making it challenging to maintain their effectiveness over time.
(iii) limited examination or discussion on the generalization of proposed solutions to new phishing samples.

1.1. Overview and Motivation

In this thesis, we aim to build an anti-phishing system using different data modalities, employing two subfields of artificial intelligence: natural language processing for textual data and computer vision for images. Textual data encompass various features integrated from the website's core content, including the website's URL and the main content extracted from the website's source code. Additionally, screenshot images convey the web page's visual appearance which contains discriminative features that would indicate whether a web page is a phishing or a legitimate one.

1.1.1. Research Questions

In this thesis, we attempt to address the following research questions:

RQ1: Are existing anti-phishing approaches effective and sustainable, what data sources are they employing, and do they possess limitations?

RQ2: Are multimodal anti-phishing datasets available, incorporating raw data sources? If so, what data sources do they include?

RQ3: How does combining textual and visual data sources impact the detection system's performance? How does it compare to single-modal alternatives in terms of generalizability?

1.2. Contributions

In this thesis, we propose a novel, heterogeneous, multimodal, end-to-end, deep learning model named *CrossPhire* to classify phishing and legitimate web pages by fusing their visual and semantical information. We follow an interdisciplinary approach by leveraging state-of-the-art models from natural language processing and computer vision fields. *CrossPhire* employs state-of-the-art models to operate on the web page's distinct data modalities (i.e., URL, image, and text) in a parallel fashion. Hence, it represents the different aspects of a web page by considering visual (screenshot image) and textual information (URL and source code). This data-informed approach enhances the model's decision-making capabilities by capturing deep holistic visual features and informative, contextual, and semantic textual relationships. Another novelty in this thesis is the utilization of state-of-the-art Sentence Transformers (ST) on the web page's extracted core content in the scope of web page phishing detection. We consider various design options by experimenting with different ST models, namely Sentence-BERT and multilingual XLM-RoBERTa on English and non-English text extracted using various HyperText Markup Language (HTML) parsing tools. We experimented with state-of-the-art deep CNN models, namely DenseNet121 and ResNet50, and fine-tuned them on web page screenshot images for our phishing detection task. Lastly, we consider URL-based phishing detection models for URLs, namely, GramBeddings, URLNet, and URLTran. Combining the invaluable insights from our extensive experiments, we propose the new architecture *CrossPhire*, by combining sentence transformer and Deep CNNs models for web page phishing detection.

Our approach sets itself apart from other methodologies by: (i) Providing a language-agnostic solution by fusing the web page's primary data modalities (URL, Image, HTML source code), (ii) Fine-tuning pre-trained state-of-the-art NLP and computer vision models specifically for the phishing detection task and utilizing them to capture complex temporal, visual, and semantical relationships, (iii) Providing robustness against zero-day attacks while maintaining autonomy from third-party features, (iv) Mitigating the data leakage problem created by duplicate samples in anti-phishing datasets by employing a multimodal approach, and (v) Contributing a novel multimodal architecture that is robust against evasion and obfuscation techniques.

The main contributions of this thesis can be summarized as follows:

- We propose a novel heterogeneous data-informed approach fixated on extracting meaningful, discriminative, and holistic features from raw data modalities named *CrossPhire*. The inherent multimodal nature of *CrossPhire* empowers the collective ability to distinguish phishing and legitimate web pages. Utilizing state-of-the-art models capable of capturing and representing complex nuances between phishing and legitimate samples.

- We conduct a detailed, thorough, and comprehensive literature review emphasizing the recent novel approaches utilizing different data modalities while shedding light on their advantages and drawbacks. Consequently, identifying and filling several gaps in the anti-phishing research domain.
- We conduct a detailed exploratory data analysis on several benchmark anti-phishing datasets, highlighting their strengths and weaknesses. We examine and report our observations using various statistics.
- We present the *Phish360* dataset by employing a qualitative and systematic approach in sample collection. Ensuring the quality, diversity, and inclusivity of real-world phishing samples in different languages. Additionally, Phish360 encompasses diverse samples that include a wide range of phishing tactics while ensuring the uniqueness of included samples.
- We obtain state-of-the-art results employing Transfer Learning by fine-tuning pre-trained state-of-the-art models from computer vision and natural language processing fields to the phishing identification task. Demonstrating that the employed models are capable of capturing the permanent characteristics and intricacies of phishing web pages.
- We employ several benchmarking datasets to evaluate *CrossPhire* on samples collected in different time periods to investigate the cross-dataset generalization performance of *CrossPhire*. On top of that, we conduct extensive experiments to investigate the performance of the single-modal components, validating the design choice of our proposed scheme.

- We share our codebase and all the supplementary materials, including our Phish360 dataset, with the research community. Enabling dataset benchmarking and fair comparison for future studies employing different features.

1.3. Organization

The organization of the thesis is as follows:

- **Chapter 1:** introduces the thesis, presents our motivation, contributions, and defines the scope of this thesis.
- **Chapter 2:** provides a background overview of the crucial concepts that we address in this thesis.
- **Chapter 3:** provides a comprehensive literature review on recent web page anti-phishing solutions, highlighting their advantages and limitations. In this chapter, we categorize the reviewed studies into five main categories based on the data modality employed. We also summarize key studies from every subsection in separate tables, showcasing important information like the size of the dataset employed and the achieved performance.
- **Chapter 4:** investigates the quality and diversity of public multimodal anti-phishing datasets and introduces our new multimodal dataset *Phish360.* We explain the collection and sample selection process, showcasing the importance of the dataset by conducting a detailed Exploratory Data Analysis. We also demonstrate the importance of *Phish360* by comparing it to the existing multimodal anti-phishing datasets.
- **Chapter 5:** provides a detailed description of the proposed methodology. We first present an overview of the proposed scheme, *CrossPhire*, and then we explore each component in separate subsections. Finally, we define the evaluation metrics.
- **Chapter 6:** presents our experimental results and corroborates our design options by examining the results using different HTML parsers, Sentence transformers,

and image models. We validate our proposed architecture by conducting several ablation studies exploring the various components in CrossPhire. Additionally, we analyze and compare CrossPhire's performance using both multilingual and monolingual English texts extracted using different HTML parsers. Finally, we address several evaluation questions regarding the impact of multimodality on detection performance, CrossPhire's generalization capability by conducting extensive out-of-sample evaluations, and a comparison between CrossPhire and its baseline approaches.

- **Chapter 7:** explores the adaptability of our phishing detection framework and its capability to adapt and integrate newly emerging models to process the different data modalities. We highlight the effectiveness of leveraging the website's textual content by visualizing the textual embeddings. Finally, we discuss some limitations and drawbacks of our proposed methodology.
- **Chapter 8:** concludes the thesis and provides possible future work.

2. BACKGROUND OVERVIEW

2.1. Types of Phishing Attacks

As we discussed earlier, phishing is the act of deceiving people or entities to gain information that can be used for financial benefits, yet this is a very general definition. There are multiple types and methods where the phishers exploit different weak points to achieve their goals. These types widely vary based on the techniques used to deceive but all share the same aim which is to obtain valuable data that can be used against the victims.

Attackers tend to exploit the weakest link in the security chain, i.e., the user, as it is easier for attackers with no strong technical skills to exploit human vulnerabilities by deceiving them rather than exploiting some system's vulnerabilities [33]. To do that, the attackers rely on some form of communication, like emails, messages, or social networks, to contact the victims. The attackers must impersonate a trusted entity such as the bank's customer services or a famous and trusted service provider like Apple or Microsoft for the victim to believe them. A trick attackers use to look and feel legitimate when communicating with victims using email is to spoof the source's email address and add the appropriate company logos to look legitimate. They might use 'security update', 'password change notice', or 'urgent' in the subject line to grab the victim's attention, and when the victim checks the sender's email, it will look legitimate since the email spoofed. From there, they can redirect the victim to another fake website (phishing website) that the attackers have prepared to collect the victim's sensitive information [34]. Since this thesis focus on the deceptive types of phishing, here are some of the well-known and frequently employed types of phishing attacks:

- **Spear Phishing:** In this attack, the attackers aim to infiltrate organizations or institutions by finding inside information or social data (which could be acquired from social media such as Facebook and LinkedIn) on some users in the company. This information will be used to craft an email that appears to be from a trusted source by the user, like a supervisor or IT department employee [35]. Since the victim thinks he knows the sender's identity, the victim would not be as skeptical, and the message's content would not raise any suspicion [36]. The high success rate of spear phishing comes from faking an email or message from a person or entity known by the victim, so the victim would not be as hesitant to provide his login credentials, for example, if contacted by the company's IT department [37] [38].

- **Man-in-the-middle attack**: The idea behind these types of attacks is that the attacker sits in the middle of client-server communication, hence the name man in the middle. By being in the middle of this two-way communication between a victim and a web application, the attacker can intercept and collect the personal information that the victim is sending to the server or web app [8] [34]. Additionally, the attacker in this scenario can sit in the middle and establish two SSL connections separately, one with the victim and the other with the real server [39]. To carry out this type of attack, the attacker has to mislead the customer to a proxy server instead of the real server; that way, he can capture the data in the middle. This can be achieved using various methods like Transparent Proxies, DNS Cache Poisoning, URL Obfuscation, and Browser Proxy Configuration [34].

- **Whaling Attacks:** The term is derived from whales within poker, which refers to big-time gamblers that spend a significant amount of money [34]. Whaling in the context of phishing refers to attacks focused and highly targeted towards high-level senior executives (such as CEO, CFO, and CTO) within an organization. Unlike the other typical phishing attacks that aims to deceive as many victims as possible, whaling attacks are meticulously planned to target a small subset of individuals. In whaling attacks, the attackers spend more time crafting the targeted message to achieve the highest probability of success in stealing the high-level executives' credentials[34].

- **Smishing & Vishing:** Smishing and Vishing attacks have recently gained vast popularity among attackers. Like email phishing, Smishing and Vishing aim to steal sensitive information from the victims. However, instead of using emails to distribute their phishing websites, attackers use mobile text messages in Smishing and voice calls in Vishing [34].

2.2. Machine Learning

Machine learning (ML) stands as a subfield within artificial intelligence (AI), comprising algorithms designed to empower computers with the ability to learn and enhance performance through experience, mirroring human-like task execution [40, 41]. These algorithms become invaluable in optimizing and automating specific tasks, eliminating the necessity for direct human intervention. These algorithms can be classified into multiple categories and types depending on the criteria used for that classification. One significant criterion is the nature of the tasks they are designed to execute, and the type of data employed for task execution. This results in two primary categories: supervised learning and unsupervised learning. In supervised learning, labeled data is utilized to perform tasks, while unsupervised learning makes use of unlabeled data, as labeled data may not always be available. In supervised learning, the objective is typically to instruct the computer to replicate a known system, providing it with labeled data for guidance. On the other hand, unsupervised learning aims to enable the computer to grasp, extract, and utilize unseen patterns without predefined guidance from labeled data [42].

The progress in machine learning can be attributed to various factors, one of which is the increased usage as individuals and organizations exploring its capabilities, giving rise to several subfields. One such subfield is natural language processing (NLP), which centers on the interaction between computers and human language [43]. This fusion of machine learning and linguistics has led to diverse applications, including but not limited to text translation, paraphrasing, and text generation [44].

Deep learning stands as another significant subfield of machine learning, particularly proficient in managing large and complex datasets. It achieves this capability by employing multiple layers to process data, allowing it to discern complex patterns and relationships inherent in the dataset. Deep learning has facilitated breakthroughs, particularly in applications related to processing videos, images, and audio [45, 46]. Also, deep learning methods have demonstrated their ability to surpass earlier state-of-the-art machine learning techniques across various fields, with computer vision standing out as one of the most prominent examples [46].

The continued advancements in deep learning have given rise to the concept of transfer learning, which involves incorporating additional sources of information beyond the usually used training data. In transfer learning, knowledge from one or more related tasks is utilized to enhance the learning process in the target task. The objective of using transfer learning is to improve the performance of the given model in the target task by leveraging insights gained from the source task or tasks in a way to mimic the human capability of transferring knowledge between tasks [47].

3. RELATED WORK

Plenty of attempts used various methods to detect and identify phishing websites. The majority of research studies have been categorizing these methods into (i) list-based, (ii) similarity-based, and (iii) machine learning-based methods. However, we categorize the methods in the literature into five main categories based on the data modality used, i.e., (1) URL-based, (2) Content-based, (3) Image-based, (4) Bimodal, and (5) Multimodal approaches. In the first three categories, the studies utilize single-modal extracted features, and the fourth category includes using a combination of modalities. However, the multimodal category combines the URL, textual content, and image features.

In this section, we present an overview of the literature research by fixating on the state-of-the-art approaches in recent years. We start by summarizing some anti-phishing approaches available in the literature according to the categorization mentioned above. Then, we select the most relevant studies and summarize them in multiple tables according to their categories, highlighting important information like main findings, used datasets, and known limitations.

3.1. URL-Based Phishing Detection

The earliest URL-based works relied on white and black lists, comparing a website's information to determine its legitimacy. Researchers have used URLs, domain names, and IP addresses for comparisons. Identified phishing websites would then be added to the blacklist [48]. The drawbacks of using such techniques are the need to maintain a database containing the lists and the need to update these lists constantly. On the other hand, heuristic-based approaches have been proven successful, especially with the recent rise of machine learning. Machine Learning approaches on URLs have been widely utilized in phishing detection. To extract extinguishable features, some researchers opted for manually extracting lexical and statistical features that rely on experts' knowledge from the URL string [23, 24], such as URL length or the count of specific characters or symbols. Others utilized automatic feature extraction using NLP methods like TF-IDF in [25, 26, 49], N-gram features [30, 50], and character or word embeddings [2, 51, 52].

3.1.1. URL Statistical & Lexical Feature Based Methods

Rao et al. [25] collected and published the CatchPhish URL dataset. The authors proposed URL-based handcrafted features, TF-IDF features (see 3.1.2.), and a combination of both. Using the handcrafted features (e.g., Ratio of hyphens in URL, num of digits in path and hostname) with the RF classifier, they obtained the highest accuracy of 94.32% on the D1 proportion of their dataset.

Korkmaz et al. [23] used the CatchPhish URL dataset published in [25] to extract URL statistical features. The authors reviewed some of the existing literature to pick these distinctive features. Consequently, they picked 58 features and narrowed them down to the top 48 URL features. These selected features include the count and presence of certain characters or words in the URL and some ratios like the ratio of digits to letters. After experimenting with eight different ML algorithms, they obtained the best performance using the RF classifier on the three portions of the dataset: D1, D2, and D3. They reported accuracies of 94.59% on D1, 90.5% on D2, and 91.26% on D3.

Similarly, Butnaru et al. [24] also obtained the highest evaluation metrics using the RF classifier but using a combination of self-collected samples and samples from the Malicious And Benign URLs dataset [53]. The authors picked 10 features from previous studies and proposed two additional similarity index features (similarity between URL's domain and subdomain. domain and the top benign domains in the dataset). The authors argued against using balanced datasets in phishing detection systems as they do not represent a realistic scenario. When testing their proposed approach, they reported a testing accuracy of 99.29% using 380k samples with around 80% legitimate URLs. Finally, the proposed system was evaluated over time using samples from PhishTank [54], achieving around 94.5% accuracy.

A new concept called intra-URL relatedness was introduced by Marchal et al. [55]. This concept measures the relation between the domain and the rest of the URL string. The authors argue that while all parts of legitimate URLs are related, phishing URLs' domains are not related to their targeted brand. Including their new proposed feature, the authors used 12 total features extracted from the URL and search engine queries. The features also reflect the popularity of the URL based on third-party services like Google Trends and Yahoo Clues. The authors collected around 96k URL samples from PhishTank [54] and DMOZ [56] for phishing and legitimate samples. By experimenting with different ML models, they obtained the highest results using the RF classifier with accuracy and TPR values of 94.91% and 91.27%, respectively.

In an attempt to create a lightweight URL phishing detection approach, Haynes et al. [57] experimented with URL and HTML-based features (see 3.4.). In their URL-based phishing detection method, they experimented with deep Artificial Neural Networks (ANNs) and pre-trained transformers (see 3.1.2.1. for the transformer-based experiments). Given that reason, they collected around 22,000 phishing and legitimate URL samples from PhishTank [54] and CommonCrawl [58], respectively. Utilizing 31 URL-extracted features, they reported an accuracy of 86.2% using deep ANNs (ANNU).

Another example of URL-based lightweight approaches suitable for Internet of Things (IoT) environments was introduced by Bustio-Martínez et al. [59]. The authors proposed a system by considering some URL-based features from the literature and introducing additional novel ones. As the authors' aim is to build a lightweight phishing detection system, they employed a feature selection algorithm to reduce the number of features used. After evaluating these features, they identified and used the nine most relevant distinctive features, 6 of which were introduced by them (e.g., hostname length ratio to URL length and Entropy of the URL). Experimenting with different classification methods, they obtained the best performance using the Random Forest Classifier. They reported 99.57% accuracy using their own collected dataset containing around 52k samples from Alexa and PhishTank [54]. Finally, it is worth mentioning that the authors decided to make their dataset public and provided a link for it in their paper.

Korkmaz et al. [28] implemented a Deep Neural Network (DNN) and compared its performance with ML-based algorithms using 73 URL-based features obtained from an earlier study [60]. Some examples of these features are the number of dots in the URL path, the length of the hostname, and the length of domain name. They collected and published a 226K URL dataset (High-Risk Phishing URL Dataset on Kaggle) with equal samples from both phishing and legitimate classes. The authors reported the highest accuracy of 86.64% using DNN. Among the ML-based models, 86.62% accuracy was reported using the RF classifier with 5-fold cross-validation. In a more recent study, the same authors, Korkmaz et al. [50], utilized the self-collected High-Risk URL dataset from their earlier publication [28] to investigate the use of CNNs with URL-based n-gram features. They created character n-gram features using the values 1 to 5 for n (i.e., unigram, bigram, trigram, 4-gram, and 5-gram). The authors obtained the best performance using the CNN model while limiting the URL characters to 70. Their proposed model achieved 88.9% accuracy on URL character unigram features, and they reported a test time of 0.008 seconds for a single URL.

3.1.2. NLP Based Feature Methods

As discussed in the previous subsection, the experiments conducted by Rao et al. [25] resulted in the highest accuracy of 94.32% using handcrafted features. Combining the handcrafted features with extracted TF-IDF features, the RF's accuracy increased to 95.67% on D1. Finally, the authors validated their best-performing approach on EBBU2017 [61] and PhishStorm [55] datasets, achieving accuracies of 98.04% and 98.57%, respectively.

In 2019, Sahingoz et al. [61] introduced the Ebbu2017 dataset containing around 73k URL samples obtained from PhishTank [54] and Yandex. Using their proposed URL dataset, the authors extracted words from the URLs to create a word list. The created word list is then analyzed to filter out random words consisting of meaningless characters and symbols. First, 40 NLP-based features were extracted, like the average adjacent word length and consecutive repetition of a character. Then, weka's StringtoWordVector method was applied to obtain word vectors (102 features). After merging the features, they were reduced to a total number of 104 hybrid features. The proposed approach was validated with seven different ML models on the different features. Obtaining the highest accuracy of 97.98% using the RF classifier on the 40 NLP-based features.

3.1.2.1. Character and Word Embeddings Based Methods The study we explored earlier in subsection 3.1.1. (Statistical & Lexical URL features) also included the use of pre-trained transformers on the strings of URLs [57]. The authors fine-tuned BERT and ELECTRA models on the collected URLs. By experimenting with different variations of pre-trained BERT and ELECTRA, they reported the highest accuracy of $\sim$96% using the ELECTRA-base model fine-tuned on their collected URLs. Finally, their experiments on both URL and HTML modalities are discussed in 3.4..

Similarly, Jishnu and Arthi [51] used BERT to provide a phishing URL detection approach. The approach combines handcrafted features with BERT embeddings by concatenating them. After generating the input features, fine-tuning is applied to the pre-trained BERT model with a classification layer. Using a collection of 200,000 URLs with a 1:1 class ratio, their approach obtained 97.32% accuracy and F1-score.

In another study, the same authors proposed a phishing detection approach by employing RoBERTa for URL feature extraction and LSTM for classification. [52]. Their proposed approach leverages pre-trained RoBERTa to create contextualized embeddings by adding LSTM and FC layers. The proposed approach was evaluated on 300,000 URL samples with a 1:1 class ratio, achieving 97.14% accuracy.

Arguing against the use of handcrafted and third-party dependent features, Bozkir et al. [2] proposed a new deep neural network based on URL character-level n-grams called GramBeddings. The authors argued that there is a desperate need for bigger and more balanced URL datasets in the anti-phishing domain. Hence, they collected and published the GramBeddings URL dataset comprised of 400k legitimate from the web, and 400k phishing URLs from PhishTank and OpenPhish. The proposed model (GramBedding) employs an n-gram selection method paired with CNN, BiLSTM, and attention layers. Through extensive validations on their self-collected URL dataset, they obtained an accuracy of 98.27%. Additionally, GramBeddings was evaluated on seven different public benchmarking URL datasets, outperforming the proposed methods by the authors, achieving a minimum accuracy of 98.32% accuracy (see table 3.1). It is worth noting that, in this thesis, we utilize the GramBedding model in our proposed three-modality approach, and it will be further explained in The Proposed Approach section.

Table 3.1 Summary of URL-based anti-phishing approaches

Ref	**Dataset**	**Used Algorithm**	**Evaluation**	**Limitation**
[55] 2014	Self-collected PhishStorm (P: ∼48k L: ∼48k)	Random Forest	**Acc:** 0.9491 **P:** 0.9844 **R:** 0.9127 **F1:** 0.9472	Language and third-party service dependency.
[61] 2019	Self-collected Ebbu2017 (P: ∼37k L: ∼36k)	Random Forest	**Acc:** 0.9798 **P:** 0.970 **R:** 0.990 **F1:** 0.980	NLP features are dependent on (1) experts' knowledge, (2) English language, and (3) third-party services (Alexa).
[25] 2020	Self-collected CatchPhish (P: ∼122k and L: ∼171k) evaluated on D4: [61] and D5:[55]	Random Forest	**Accuracy:-** **D1**: 0.9567 **D2**: 0.9395 **D3**: 0.9426 **D4**: 0.9825 **D5**: 0.9857	TF-IDF is language specific. third-party service dependency (brand names from PhishTank).
[23] 2020	Used CatchPhish dataset	Random Forest	**D1:** **Acc**: 0.9459 **P**: 0.9450 **R**: 0.9469 **F1**: 0.9459 **D2:** **Acc**: 0.905 **D3:** **Acc**: 0.9126	Using handcrafted URL-based features. Lower accuracy compared to the authors of CatchPhish [25].

cont'd

Table 3.1 – continued from previous page

Ref	**Dataset**	**Used Algorithm**	**Evaluation**	**Limitation**
[28] 2020	Self-collected High-risk URL (P: ∼113k L: ∼113k)	ML vs DNN	**RF:** **Acc:** 0.8710 **P:** 0.8605 **R:** 0.8807 **F1:** 0.8705 **DNN**(M6): **Acc:** 0.8664	Comparatively low accuracy and usage of handcrafted features.
[24] 2021	Used Kaggle dataset [53] and some collected samples. Total: (P: ∼135k L: ∼364k)	Random Forest	**Acc:** 0.9929 **P:** 0.9740 **R:** 0.9906 **F1:** 0.9822	Using handcrafted URL-based features. Some features depend on legitimate domain lists. When evaluated on PhishTank over time, accuracy drops to ∼94%.
[50] 2021	Used High-risk URL dataset	CNN	**Acc:** 0.889 **P:** - **R:** - **F1:** -	Comparatively low accuracy.
[59] 2022	Self-collected URL samples (P: ∼26k L: ∼26k)	Random Forest	**Acc:** 0.9957 **P:** 0.9947 **R:** 0.9957 **F1:** 0.9968	handcrafted features and dependency on experts' knowledge.
[2] 2023	Self-collected Grambeddings (P: 400k L: 400k) used seven public datasets for evaluation	proposed DNN architecture (CNN, BiLSTM, and attention layers)	**Acc:** 0.9827 **P:** 0.9894 **R:** 0.9759 **F1:** 0.9826 **Eval Acc:** on [61]: 0.9914 on [48]: 1.00 on [55]: 0.9832 on [62]: 0.9873 on [63]: 1.00 on [53]: 0.9982 on [64]: 0.9982	The use of n-grams from URLs does not capture semantics [2].
[51] 2023	Self-collected URL samples (P: 100k L: 100k)	fine-tuned BERT	**Acc:** 0.9732 **P:** 0.9686 **R:** 0.9780 **F1:** 0.9733	The impact of handcrafted features' on the overall performance was not examined.

cont'd

Table 3.1 – continued from previous page

Ref	Dataset	Used Algorithm	Evaluation	Limitation
[52] 2023	Self-collected URL samples (P: 150K L: 150K)	RoBERTa and LSTM layer	**Acc:** 0.9714 **P:** 0.9716 **R:** 0.9711 **F1:** 0.9714	shortened URLs may pose a challenge. No recorded training time.

3.2. Content-Based Phishing Detection

Benavides-Astudillo et al. [65] investigated the use of textual content features extracted from the website's HTML code. They aim to accurately detect phishing websites using textual semantical features with DL algorithms. To do that, they used a public dataset called Phishload; after filtering out the invalid samples, they were left with 9,198 phishing and 1,176 legitimate samples. The authors first used Regular Expressions to remove HTML tags, digits, and junk characters from the HTML source code. They then applied some NLP preprocessing steps, e.g., stop-word removal, tokenization, and lemmatization. They used the Keras Embedding layer and GloVe word embeddings for feature representation. After experimenting with different maxlen values to determine the maxlen of the string, the authors chose 200. Finally, they tested four different DL algorithms and reported a mean accuracy of 97.39% with 5-fold cross-validation using the Bidirectional Gated Recurrent Unit (BiGRU).

Arguing against using handcrafted and manual feature extraction methods, Opara et al. [66] proposed a DL-based approach called HTMLPhish. The proposed approach concatenates word and character embeddings generated using the Raw HTML content. In the preprocessing steps, the Raw HTML content is first tokenized and then padded (The dimension for each vector is set to 100, maxlen for characters is 180 and 2000 for words) before it is passed to the embedding layers. It is worth noting that HTML tags, punctuations, and symbols were not removed from the HTML content. After separately creating word (HTMLPhish-Word) and character (HTMLPhish-Character) embeddings, they employed a dense layer for concatenation. The authors collected two separate datasets, D1 and D2: D1 was collected in Nov 2018, consisting of 23k legitimate and 2.3k phishing samples.

D2 was collected in Jan 2019, composed of 24k legitimate and 2.4k phishing samples. In the evaluation step, HTTMLPhish-Full, HTTMLPhish-Word, and HTMLPhish-Character were trained using D1 and tested on the D2 dataset. HTMLPhish-Full performed the best, achieving a 93% testing accuracy, while HTMLPhish-Word and HTMLPhish-Character achieved 90% and 91% testing accuracies, respectively.

Arguing against handling the HTML source code as a sequence of characters to extract representative features, (Ouyang and Zhang) [21] proposed a Graph Neural Network-based approach. The authors first parsed the HTML source code into DOM trees where tags and inner tags are represented using nodes and edges, respectively. The authors used an RNN to extract feature vectors from the node attributes. Max pooling was employed to ensure the acquisition of fixed-length RNN output vectors. Additionally, the Topology Adaptive Graph Convolutional Network (TAGCN) was used to capture the long-rage semantics between the nodes in the graph. Finally, all embeddings were reduced using max-pooling and passed to the fully connected layer for classification. Their experimental results on a self-collected 121,983 legitimate and 26,578 phishing samples achieved 95.5% accuracy.

Although Ariyadasa et al. [67] used both URL and HTML features to detect phishing sites in their proposed approach PhishDet (see 3.4.). PhishDet consists of URLDet and HTMLDet, handling the two different data modalities. HTMLDet is a representation learning approach based on Graph Convolutional Networks (GCN). The HTML content of the webpage containing tags is used to construct the graph structure, and the vectors for each document (feature matrix) are generated using Doc2Vec. The HTMLDet approach trained and validated using the HTML modality alone obtained 89.87% accuracy.

Finally, proposing a new phishing detection approach, Rao et al. [68] extracted plain and domain-specific text from the web page's source code. As the domain-specific extracted text is concatenated with URL-based features, we only highlight the authors' experiments using the extracted plain text. The BeautifulSoup parser is first used to extract the text from the HTML source code. Then, the extracted text is tokenized and lemmatized. The authors used several word embedding generation methods for feature representation, achieving the best results using the FastText model with Skipgram. Evaluating their proposed approach on 5,438 phishing and 5,076 legitimate samples with different ML algorithms, they obtained 98.28% accuracy using the Logistic Regression model.

Table 3.2 Summary of content-based anti-phishing approaches

Ref	**Dataset**	**Used Algorithm**	**Evaluation**	**Limitation**
[66] 2020	Self-collected ∼52k HTML samples (**D1** = P: 2.3k L: 23k **D2** = P: 2.4k L: 24k)	Deep Learning (CNNs)	HTMLPhish -Full **D2:** **Acc: 0.93** **P:** 0.92 **R:** 0.93 **F1:** 0.91	The used model doesn't fully represent the semantical relationship of the text (representation of misspelled words will be similar to the actual word). And word-level embeddings depend on the training data [27]. The performance declines with time and needs a retraining phase [67].
[69] 2020	Self-collected samples combined with PWD2016 [32] (P: 20k L: 20k)	CNNs	**HTML only**: **Acc: 0.9170** **P:** - **R:** - **F1:** -	The performance of the single modal approach (HTML only, referred to as 1D Conv in [69]) is comparatively low. Manual feature extraction (handcrafted).
[21] 2021	Self-collected ∼149k samples (P: ∼27k L: ∼122k)	Graph Neural Network (GNN) and RNN	**Acc: 0.9550** **P:** 0.9345 **R:** 0.8025 **F1:** 0.8634	The method can be bypassed by cloning legitimate HTML code structure [22].
[68] 2022	Some self-collected samples (P: ∼5.4k L: ∼5k)	Logistic Regression	**EX1-PT** **Acc: 0.9828** **P:** 0.9787 **R:** 0.9878 **F1:** 0.9832	The proposed approach fails when the text is replaced with an image [68]. The dataset contains only English samples.
[67] 2022	Some self-collected samples with PWD2016 [32], web2vec [70], and [71] (P: ∼66k L: ∼70k)	(HTMLDet) Graph Convolutional Network (GCN)	HTMLDet **Acc: 0.8987** **P:** 0.9118 **R:** 0.8828 **F1:** 0.8971	Comparatively low accuracy

cont'd

Table 3.2 – continued from previous page

Ref	Dataset	Used Algorithm	Evaluation	Limitation
[65] 2023	Used PhishLoad (P: ∼9k L: ∼1k)	BiGRU	**Acc: 0.9739** **P:** - **R:** - **F1:** 0.94	Limited and unbalanced dataset with ∼2k validation samples. Limited the maxlen value to 200 (L= max number of words).
[27] 2024	Self-collected legit samples. Phishing samples from High-risk dataset [28] (P: ∼23k L: ∼23k)	CNN (referred to as Option3 using only HTML)	WebPhish (O3) HTML **Acc: 0.965** **P:** 0.970 **R:** 0.961 **F1:** 0.965	The proposed model would need retraining to stay relevant and maintain high performance and the model cannot classify webpages that replace DOM with embedded images [27].

3.3. Image-Based Phishing Detection

(Phoka and Suthaphan) [20] proposed employing pre-trained CNNs on images of a login web page in phishing detection by considering five different targeted brands. They aim to conduct two main experiments: Binary and Multiclass classification. The authors proposed a data augmentation method to enlarge their dataset by producing new sample images. Their method consists of sub-image identification and sub-image random placement. It is done by identifying regions of interest on a login page and replacing them with other randomly identified sub-images from the same sample. By fine-tuning five different pre-trained CNNs (Inception V3 & V4, ResNet V1 & V2, and ResNet-Incepttion) on their produced data samples, they obtained the highest accuracy of 97.1% using the Inception-ResNet-v1 model.

(Bozkir and Aydos) proposed a vision-based phishing detection approach using logo detection called LogoSENSE [19]. The web page's logo is detected using Max-margin objection detection (MMOD). The detected logo is used to extract features using the Histogram of Oriented Gradients (HOG) descriptors. The generated HOG vector contains visual information obtained using a logo-sized sliding window on the screenshot. The resultant HOG vectors are classified based on their similarity with the targeted brand logos. LogoSENSE was validated on a self-collected image dataset containing 3,060 training and 1979 testing image samples, achieving 93.50% precision and 85.02% F1-score.

Addressing the unexplainable classification results of phishing detection systems, Lin et al. [18] proposed a deep learning vision-based approach called Phishpedia using brand logos identity. Phishpedia is a two-step solution where, given a screenshot, Phishpedia first detects the logo and then matches it to a reference of legitimate brand logos. The Faster R-CNN is used to recognize and detect the logos and input boxes (forms) from a webpage using their self-collected and annotated 30,649 legitimate samples. The authors trained the Faster R-CNN model by jointly training the region proposal network and Fast-RCNN on the extracted feature maps. In the brand recognition task, the authors opted for a Siamese model employing ResNet architecture to identify the logo's brand. The ResNet backbone model was pre-trained on the Logo2k+ and then fine-tuned on their self-collected samples consisting of 181 targeted brands' logos. To ensure that the features of dissimilar logo variants of the same brand identity are represented, they connected the backbone model with a global average pooling layer. Evaluating Phishpedia on their self-collected dataset, they outperformed relevant state-of-the-art approaches, obtaining a detection rate of 89.2% precision, 87.1% recall, and a phishing identification rate of 99.2%.

Table 3.3 Summary of image-based anti-phishing approaches

Ref	**Dataset**	**Used Algorithm**	**Evaluation**	**Limitation**
[20] 2019	Self-collected and augmented image samples (P: - L: -)	pre-trained CNNs (Inception- ResNet-v1)	Binary class. **Acc: 0.971** **P:** - **F1:** -	The method does not consider highly dynamic pages [20]. The dataset size is not mentioned, and no real phishing samples were employed. The experiment only considers 5 brands.
[19] 2020	self-collected LogoSENSE dataset (P: $\sim$2.5k L: $\sim$2.5k)	Histogram of Oriented Gradients (HOG) Support Vector Machines	**Acc:**- **P:** 0.9350 **R:** 0.7794 **F1:** 0.8502	Generalization capability is challenged with unseen screenshots [18]. Limited dataset and comparatively low performance.
[18] 2021	self-collected Phishpedia dataset (P: $\sim$k L: $\sim$k)	Siamese model (brand identification) and Faster-RCNN (Object Detection, OD)	Identification Rate: 0.992 **Acc:**- **P:** 0.982 **R:** 0.871 **F1:** - **OD mAP:** 59.7	Phishpedia cannot identify phishing websites impersonating new legitimate brands (not in the reference list). Legitimate pages having a logo similar to famous brands would be misclassified as phishing [18].

3.4. Bimodal Phishing Detection

Ariyadasa et al. [69] proposed a hybrid anti-phishing solution utilizing LSTM and CNNs using URLs and HTML-based features. For the HTML features, the authors selected 15 features that are extracted from the HTML source code, such as the number of hyperlinks, length of the HTML code, the count of some HTML tags (<script>, <link>, and <!->), and a binary feature indicating the presence of the favicon. The URL characters are converted into integer values and the URL's longer than 150 are truncated. To process the URLs, the authors implemented a feed-forward network containing convolution and LSTM layers, referred to as model A. Model B on the other hand, consists of 1D convolutional layers that handle the HTML features. The authors evaluated their proposed approach on 40,000 self-collected samples with a 1:1 class ratio, achieving 98.34% accuracy. However, the main limitation of this approach is the dependency on manually extracted HTML features.

The solution proposed by Ariyadasa et al. [67], PhishDet, is a hybrid technique with both URL and HTML features to detect phishing websites. The authors proposed a DL architecture consisting of two networks trained separately on the two data modalities, then combined using a concatenation layer. The first one is Long-term Recurrent Convolutional Networks (LRCNs) for URLs referred to as URLDet. The second one is a Graph Convolutional Network (GCN) referred to as HTMLDet for the HTML content (discussed in 3.2.). The authors used three different datasets to train and validate their proposed approach. Dataset A, consisting of self-collected samples as well as samples from the PWD2016 dataset [32], dataset B [71], and web2vec dataset published by Feng et al. [70] (referred to as benchmark dataset in PhishDet). Dataset A was used for the initial training, and then their proposed approach was retrained on dataset B. When evaluated on dataset B, PhishDet achieved 96.42% accuracy and F1-score.

Another example of an end-to-end DL-based network is WebPhish. WebPhish is a hybrid approach using raw URL and HTML content proposed by Opara et al. [27]. The authors proposed an end-to-end DNN approach using CNNs, creating character-level embeddings from URLs and word-level embeddings from raw HTML. The URL preprocessing steps include URL protocol removal (HTTP:// and HTTPS://) and splitting by characters, creating a dictionary of 75 unique letters. The Raw HTML content is tokenized into words, considering punctuations as separate tokens, turning into a dictionary of 321,009 unique words. The URL strings are truncated at 180 characters, creating a [180, 16] embedding matrix, and the raw HTML tokenized words are truncated at 2,000 words, creating a [2000, 16] embedding matrix. The authors picked the max length values for URL and raw HTML strings by examining their self-collected dataset. The collected dataset consists of 45,373 samples of URLs and HTML content and was published on Kaggle [72]. The embeddings from the two different modalities are then concatenated into a matrix with [2180, 16] dimensions. Their evaluations on the collected dataset resulted in a 98.1% for both accuracy and F1 score.

Haynes et al. [57] experimented with URL modality using two different methods in 3.1.1. and 3.1.2.1., then combined both URL and HTML modalities to investigate their performance. For this task, they used an extended version of a dataset produced by Shirazi et al. [73], consisting of 48 URL and HTML-based features. They trained and tested the proposed deep ANNs (ANNF) on the extended dataset, reporting 0.97 normalized true-positive and true-negative values, respectively. It is worth noting that combining both modalities (ANNF) achieved higher results (∼0.972 accuracy) compared to the handcrafted experiments, i.e., ANNU in 3.1.1. and ELECTRA in 3.1.2.1..

Van Dooremaal et al. [31] proposed a hybrid phishing detection approach utilizing text and visual features. The authors collected and published a multimodal dataset. However, in their proposed approach, they randomly sampled 2,000 phishing and legitimate samples with a 1:1 class ratio. The authors proposed using the HTML web page title as a textual feature. To identify the targeted webpage, the authors first locate image regions that contain identifiable information to determine the targeted brand. These image regions are then used in a reverse image search to obtain visually similar associated web pages. To identify these regions, Otsu's thresholding algorithm, morphological closing, and topological structural analysis were employed. As the image regions may contain non-identifiable information, a filtering process was applied to remove these regions based on multiple variables like region dimensions and Dominant Color Percentage (DCP). This Target identification approach was validated on 2k phishing and legitimate samples with a 1:1 class ratio, obtaining 99.2% and 92.0% accuracies for phishing and legitimate target identification, respectively. After identifying the targeted webpage for a given sample, the visual similarity between the two images is computed using various approaches. Logistic regression was employed to compute the threshold similarity value for classification. Validating their classification approach on the 2k filtered samples, the authors achieved 99.66% accuracy and 99.77% F1-score using the Structural Similarity Index Measure (SSIM) paired with the DCP filter.

Sánchez-Paniagua et al. [30] proposed a phishing detection approach using URL, HTML, and web technology features. The authors highlighted the importance and the lack of incorporation of website samples containing login forms. For that reason, a multimodal Phishing Index Login Websites dataset (PILWD-134K) containing 134,000 samples was collected and published. Utilizing their collected dataset, they proposed 27 novel handcrafted features and adopted another 27 features from the literature. Experimenting with different ML models, the highest accuracy of 97.95% was obtained using the LightGBM classifier on the full 54 features. However, the proposed approach relies on handcrafted features that are tedious and require significant human effort. It is worth noting that we utilize the PILWD134k dataset (denoted as D1 in their original paper) in this thesis.

Liu et al. [29] proposed a hybrid reference-based phishing detection approach called PhishIntention by visually extracting the targeted brand and the webpage's credential-taking intention. PhishIntention captures the brand and credential-taking intention using a combination of deep learning vision models. PhishIntention first locates salient rectangular regions like logo, button, and input box from the screenshot image, then extracts the Abstract Webpage Layout (AWL) of the webpage. The authors employed an OCR-aided brand recognition based on Siamese matching using the extracted logos by concatenating embeddings from ResNetV2-50 and ASTER encoder. This logo-matching scheme was trained and validated on Logo2k+, Synth90k, and SynthTesxt, achieving 89.1 % accuracy. The classification of the Credential Requiring Page (CRP) is achieved using the screenshot and AWL of the webpage with ResNetV2-50, reporting a 95.0% prediction accuracy.

The authors define CRP transition as a link or a button that lures victims into another webpage (CRP). CRP transition regions are located by first examining the HTML to find clickable DOM elements with selected keywords (i.e., English or non-English text that translates to login and sign up), and if this method fails to locate these elements, then a visual object detection method is employed. Treating it as an object detection method, the authors used a Faster R-CNN to predict the CRP transition regions by fine-tuning it on labeled screenshots. Combining visual and HTML methods to locate CRP transitions resulted in a 93.3 % accuracy. Some limitations of PhishIntention include the inability to detect phishing webpages that use a new form of UI component to steal credentials like QR codes. As PhishIntention interacts with the webpage to verify the credential-requiring intention, CAPTCHA systems will challenge and block that interaction. Finally, it is worth noting that in this thesis, we employ the PhishIntention multimodal dataset published by the authors containing 29,496 phishing and 28,449 legitimate (Benign & Misleading) samples.

3.5. Multi-Modal Phishing Detection

Back in 2019, (Rao and Pais) [74] developed a search engine-based phishing detection system, Jail-phish, addressing the issue of Phishing Sites Hosted on Compromised Servers (PSHCS). They highlighted the limitations of existing search-engine-based detection systems, especially when dealing with PSHCS. Their proposed solution, Jail-Phish, extracts the domain from a given URL and the title from the webpage's source code. The domain and title are then used as a search query on Google to retrieve the first ten results. If the query domain is not in the first ten results, the sample is classified as phishing. Otherwise, the similarity between the query webpage and the results is computed using the Jaccard Similarity. The similarity between two web pages is calculated by extracting URLs, CSS, Javascript files, and images like logos or favicons from the web pages. The sample is classified as legitimate if the similarity score is greater than zero and phishing otherwise. Validating their proposed approach on a self-collected 6,067 legitimate and 5,384 phishing samples, Jail-Phish obtained 98.61% accuracy. However, Jail-Phish depends entirely on the search engine and the query results. Additionally, Jail-Phish would fail to detect the phishing site, PSHCS in particular, if the phishing site is indexed by the search engine.

Yu et al. [75] analyzed phishing websites to extract a few self-defined (handcrafted) features from the URL, HTML, and images. Paired with that, they used a multilayer perception, achieving 92.75% accuracy using the selected 11 statistical features alone. Next, they processed the HTML by retaining the tag parts and then tokenized the processed HTML and URL strings, building two separate lexicons. Followed by setting a maximum length of 16 and 256 for URL and HTML words. Using the count occurrences from the created lexicons, they transformed URL and HTML text into numbers to be fen into the LSTM layers. The screenshot images were preprocessed using four steps: (1) Image binarization, (2) Morphological closing, (3) Topological structural analysis, and (4) filtering out images less than 20 in width or length and greater than 400 × 400 regions. A CNN with CBAM attention layers was employed for image feature extraction. The authors collected 6,000 samples with a 1:1 class ratio to evaluate their proposed approach. The abovementioned extracted vectors are then concatenated and fed into the fully connected neural network. Combining the three different modalities, their proposed architecture obtained 97.75%, 96.65%, and 97.82% for accuracy, precision, and F1-score, respectively. Finally, it is worth noting that they also experimented with combinations of the different modalities (Table 3.4) and obtained accuracies of ∼93% and ∼96%.

Lin et al. [76] proposed a multimodal approach utilizing the three main data modalities (URL, HTML, and screenshots) called SenseInput. The authors utilized 22 features (nine novel features) divided into Statistical and Sensitive input features. The statistical features include URL length, digit counts in domain and subdomain, and novel features like the number of <input> tags in the HTML and the number of tags with class attributes. The sensitive input features are (1) the existence of sensitive inputs and (2) the existence of sensitive information. They first detected the sensitive input using the Faster-RCNN model on the screenshot images to acquire the sensitive input features. Then, they extracted the detected sensitive text using EasyOCR [77], a ready-to-use OCR model supporting various languages. The authors collected and published their dataset containing ∼6k phishing and $\sim$42k legitimate samples of URL, HTML, and screenshot images. They also utilized $\sim$25k phishing and around 24k legitimate samples from the Phishpedia dataset [18]. For their proposed sensitive input detection, 1k phishing and 4k legitimate screenshots were annotated. Validating the sensitive input detection on $\sim$8k collected samples and $\sim$29k samples from Phishpedia resulted in 96.94% and 96.73% F1-scores, respectively. To classify the websites using their proposed 22 features, the authors used the LightGBM classifier, reporting 98.48% F1-score on 1k self-collected samples and 95.87% F1-score on $\sim$48k samples from Phishpedia.

Highlighting the limitations of existing works on identity-based detection, Tan et al. [78] extended the work by introducing a new detection approach that leverages joint visual and textual identity. For the *textual identity discovery*, the authors first extract identity keywords from the URLs and plain text to be used as a textual reverse search query. The employed identity keywords extraction approach is adopted from their earlier work, PhishWHO [79]. PhishWHO is a three-step phishing detection approach using (1) a weighted URL token system and N-gram word extraction from relevant HTML tags, (2) finding the target domain name using a reverse search engine, and (3) an identity-matching technique to determine the legitimacy. The *visual identity*, on the other hand, is determined by first detecting the web page's logo and extracting it to be used as a reverse image search query. To detect and extract the web page's logo, the authors propose a novel logo detection technique inspired by human vision. First, a logo candidate collection is identified by first rendering the webpage in a browser and examining the DOM's elements that potentially contain an image (excluding the area outside the viewport). The number of obtained image candidates is then reduced by applying a set of filtering rules, e.g., width or height less than min 15 pixels, image height 1/3 of the image width. To decide which image is most likely to be the web page's logo, the authors picked eight features (F1 - F8) to rank the image candidates using multiple criteria decision-making (MCDM). Some of the proposed features are the vertical and horizontal position of the image, the padding space between the actual logo and the edges of the image, and a novel feature called ”colourfulness” that differentiates between logo and non-logo images. After selecting the logo with the highest rank, the authors apply some post-processing steps to optimize the logo to be used in a search engine query. Moreover, the brand name is extracted from the logo image using an OCR service. In the cases where the logo is undetected due to the use of textual logos, where the logo is loaded from text instead of an image, or if a webpage does not have a logo (e.g., personal blogs or archive sites), the proposed approach relies on the textual identity of the web page. By combining Visual and Textual identity features, the returned search results are used to determine the legitimacy of the queried sample by comparing their ccTLD (country-code Top-level Domain) and IP addresses. To validate their proposed approach, the authors collected DS-1, containing 500 legitimate and 500 phishing samples, and DS-2, consisting of 250 legitimate samples, ranked 5-6k in the Alexa top website list. The authors obtained a 98.60% accuracy, 99.40 % recall, and 97.80% TNR on DS-1, outperforming their earlier work, PhishWHO. It is worth noting that the proposed approach considers web pages with no input fields harmless, thus classifying them as legitimate.

Arguing against single-modal approaches, Zhou et al. [80] proposed a new multimodal phishing identification approach based on dicision-level fusion using model stacking. The authors advocated for the integration of the three main data modalities, i.e., URL, Text, and screenshot image. Contrary to the majority of the reviewed studies that framed the problem as a binary classification problem, the authors defined 11 classes with each class comprising of 1,000 samples, resulting in a total dataset of 11,000 samples. The authors applied some pre-processing steps like word segmentation and stop-word removal, and used BERT to extract textual features using the processed text data. Additionally, they extracted six hand-crafted URL features such as the number of dots in the domain name, and URL length, which are then fed into a logistic regression model. For processing screenshot images, the authors opted for the ResNet model. The comparative analysis between the performance of single-modal components and the ensemble multimodal approach (MultiiRECG) revealed that the multimodal approach outperformed the single-modal ones by achieving an accuracy of 88.82%.

Table 3.4 Summary of multimodal anti-phishing approaches

Ref	**Dataset**	**Used Algorithm**	**Evaluation**	**Limitation**
[74] 2019	Self-collected samples (P: ∼5.4k L: ∼6.1k)	Jaccard Similarity Index	**Acc:** 0.9861 **P:** 0.9926 **R:** 0.9777 **F1:** 0.9851	Third party dependency (search engine) slows down the process. Jail-phish uses the value zero as a similarity threshold, which sometimes negatively affects the identification of phishing pages. The approach would misclassify a phishing site indexed on the search engine as legitimate [74].
[75] 2022	Self-collected samples containing URL, HTML, and screenshot (P: 3k L: 3k)	MLP, CNN, and RNN	**Full multimodal** **Acc:** 0.9775 **P:** 0.9665 **F1:** 0.9782 **HC+Image:** **Acc:** 0.9333 **HC+text:** **Acc:** 0.9616	Very limited dataset. employing hand-crafted (HC) features.
[76] 2022	Self-collected samples (D1) containing URL, HTML, and screenshot (P: ∼6k L: ∼42k) and (D2) Phishpedia [18]	Faster-RCNN for sensitive input detection and LightGBM for phishing classification	**Input det.** **D1**-F1: 0.9694 **D2**-F1: 0.9673 **Phish classi.** **D1**-F1: 0.9848 **D2**-F1: 0.9587	Using handcrafted features. Sensitive features depend on their detection and recognition accuracies.
[78] 2023	Self-collected samples (P: 500 L: 750)	Multiple Criteria Decision-making (MCDM)	**Acc:** 0.986 **P:** - **R:** 0.994 **F1:** -	The proposed approach depends on a third-party service (Search Engine), where textual and visual identity discovery is determined using image and textual reverse Google search.
[80] 2023	Self-collected samples (P: 5,000 L: 6,000)	Ensemble stacking approach using BERT, ResNet, and Logistic Regression.	**Acc:** 0.8882 **P:** - **R:** 0.8882 **F1:** 0.7949	Relatively low accuracy and hand-crafted URL feature usage.

4. DATASETS

The recent advancements in AI have shown that ML and DL approaches provide promising solutions to diverse challenges across various domains. With the rise in the number and sophistication of phishing attacks, researchers are enhancing their approaches by leveraging newly emerged methods to stay ahead of these threats. Deep learning approaches, in particular, have caught the attention of researchers due to their demonstrated superiority over conventional machine learning algorithms in different subfields of AI. Unlike ML algorithms, deep learning models can be pre-trained and fine-tuned on new data, which is more sustainable and adaptable to the dynamic nature of phishing websites. However, the absence of large standardized datasets poses a significant challenge in anti-phishing research. The rapid advancements in computer vision and NLP domains can be attributed to the large standardized public datasets like ImageNet [81], Microsoft COCO [82], and large text corpora like the SNLI Corpus [83] English Wikipedia, and BooksCorpus [84].

Although the literature is rich with anti-phishing solutions leveraging various data modalities, public datasets are deficient. Most of the studies in the literature are conducted on self-collected private datasets, with important details like the source selection, sample diversity, data content, included languages, and sample size being withheld. Therefore, researchers desperately need a clear standard that can be followed in dataset construction. Examining the available public anti-phishing datasets, we notice that there are several issues regarding:

- **Sample Size**: There is no agreement among the scholars on the ideal dataset sample size in the anti-phishing community. The number of samples employed in anti-phishing studies ranges from $\sim$100 to over a million samples [32].
- **Class distribution**: while most scholars endorse using balanced datasets due to their positive effect on model efficiency, others are employing and creating imbalanced datasets, with most samples being legitimate. They argue that imbalanced datasets

better represent the real-world distribution of phishing to legitimate websites on the internet [21], which is a 1:10 ratio [66].

- **Data source**: Many anti-phishing works use PhishTank and OpenPhish to obtain phishing samples. For legitimate samples, Alexa is frequently used to obtain popular and highly ranked websites. However, numerous studies argued that excluding less popular websites creates biased systems and leads to higher false positives. Additionally, numerous research has pointed out that ranking lists like the ones found in Alexa provide URLs in the form of domain and TLD (e.g., facebook.com), which requires additional processing steps to retrieve the full URL. Therefore, recent studies have started to utilize other sources like DMOZ and Common-crawl or create their custom crawler to diversify the samples regarding website categories and languages and to ensure unpopular websites are also included.

- **Dataset content**: public anti-phishing datasets have varying data sources without a uniform agreed-upon procedure on what type of files to store the samples in or which data source to share (raw sources or extracted features). Reviewing some public anti-phishing datasets, we noticed a significant percentage of repeated and duplicate samples. Moreover, some datasets contain dead (offline) web page samples, usually showing error messages like 'not found' or 'bad request'. These error messages indicate that the web page source code was acquired after the website was shut down. Finally, we also observed that most shared datasets lack an adequate description of the dataset content.

- **Open-sourcing**: there is currently no commonly used platform to store and share the datasets. Hence, researchers use platforms like GitHub, Mendeley, Kaggle, cloud storage, and personal/institution websites to share their collected datasets. This lack of a unified platform makes it difficult to navigate and explore the public anti-phishing datasets. We have also observed that some links to published datasets in the original publications are no longer valid due to the relocation of the datasets.

The only viable solution for the unavailability of reliable supervised training anti-phishing data is by cumulative and collaborative community efforts aimed at publishing collected datasets as well as implemented models. With several phishing verification platforms like PhishTank and OpenPhish, it is relatively easy to collect phishing URL samples even after the websites get shut down. However, given the short lifespan of phishing websites, collecting additional resources like the source code or image files while the websites are up is significantly challenging. Therefore, it is crucial to acquire all the necessary data sources from online phishing websites as soon as they are submitted.

4.1. Is There a Need for Multimodal Datasets?

Several studies have proposed some guidelines in an attempt to standardize anti-phishing datasets and the collection process [24, 32, 85]. However, the lack of general consensus on what data sources to include in datasets makes the process more challenging. Given the short lifespan of phishing websites, retrieving all the necessary raw data sources from the website before it shuts down is imperative. Therefore, datasets with single-modal data (e.g., most commonly URLs) cannot be leveraged in bimodal or multimodal anti-phishing experiments since additional data cannot be retrieved (source code or image files). Additionally, datasets containing features rather than raw data (e.g., the UCI dataset) restrict the potential usage of the dataset by restricting the extraction of new features and effectively making the dataset inextensible. Therefore, collecting at least the three main data modalities (i.e., URL, screenshot image, and HTML source code) and storing them in their raw format is necessary. That way, the collected samples can be useful and serve the different needs of future research.

After an extensive literature review, we only found four public multimodal datasets with adequate number of samples having at least raw URLs, screenshot images, and HTML source code files. Compared with the tremendous number of anti-phishing studies in the literature, having only four benchmarking datasets speaks to the severe lack of multimodal datasets. Given this limitation, we constructed and published a new anti-phishing dataset to fill this literature gap. In the upcoming subsections, we explore and analyze our collected multimodal dataset and the publicly available datasets, revealing their advantages and disadvantages.

What are the benefits of multimodal anti-phishing datasets and their advantages over single-modal datasets?

- **Benchmarking**: multimodal datasets include different raw data sources, which broadens the potential use cases by increasing the usability of the dataset. Unlike single-modal datasets, multimodal datasets can serve as benchmark datasets and be useful for various studies with different objectives and approaches (i.e., single-modal, bimodal, and multimodal schemes). Additionally, public datasets containing dated samples are essential to keep track of phishing tactics evolution over time [86].
- **Feature representation**: multimodal raw data offers a versatile and simplified solution for automatic feature extraction methods. In fact, several studies have demonstrated the superiority of automatic feature extraction to manual ones in different fields. This also enables researchers to extract and define new features for their proposed anti-phishing solutions. Moreover, recent anti-phishing studies have demonstrated that bimodal and multimodal features are more robust against evasion techniques than single-modal features. Therefore, having access to multimodal datasets makes it easier to conduct bimodal and multimodal experiments.
- **Mitigating Data leakage**: multimodal data sources are essential to mitigate the critical data leakage issue in predictive models. Duplicate samples severely limit single-modal anti-phishing datasets, as phishers use multiple URLs for the same website (detailed in upcoming sections). However, multimodal data and its triplet nature guarantee the uniqueness of phishing samples, which is critical to obtaining generalizable and accurate predictive models.

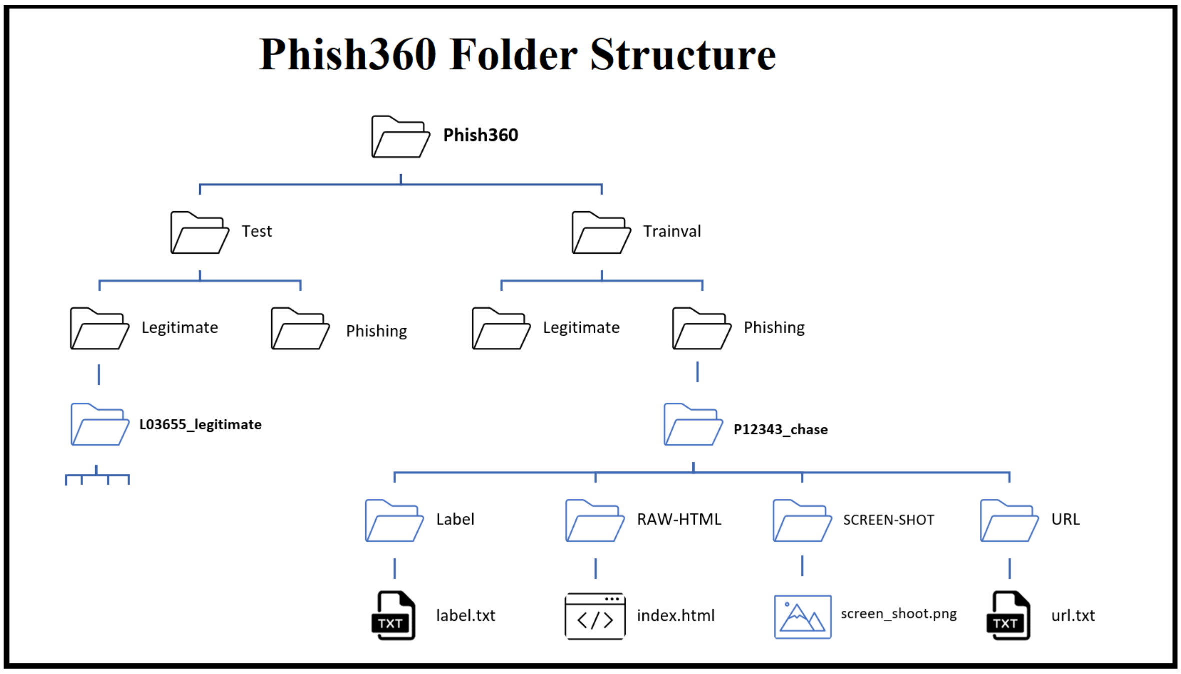


Figure 4.1 Phish360 dataset folder hierarchy showing the naming convention and the file types in the dataset

4.2. Phish360 Dataset

The collection of the dataset spanned two and a half years to ensure the variety of the samples in the dataset. We originally collected x samples, then filtered the samples and removed the duplicates, leaving us with 11,007. After further investigation, we found some samples with missing attributes, so we decided to remove such samples. The final number of samples after the removal and filtration is 10,748 samples. The URL of each sample is the final URL of the website.

This dataset is a collection of Legitimate and Phishing website samples. Each sample contains the URL, Label, HTML content, and a screenshot of the website. The Phish360 dataset contains 10,748 samples, which consist of two main categories: phishing and legitimate. There are 6,416 legitimate and 4,332 phishing samples. We divided the dataset into two main folders for training and testing using a 3:1 split (75% training and 25% testing). The folder hierarchy of the dataset is illustrated in Fig 4.1. And the class distribution in both folders is shown in Fig. 4.2.

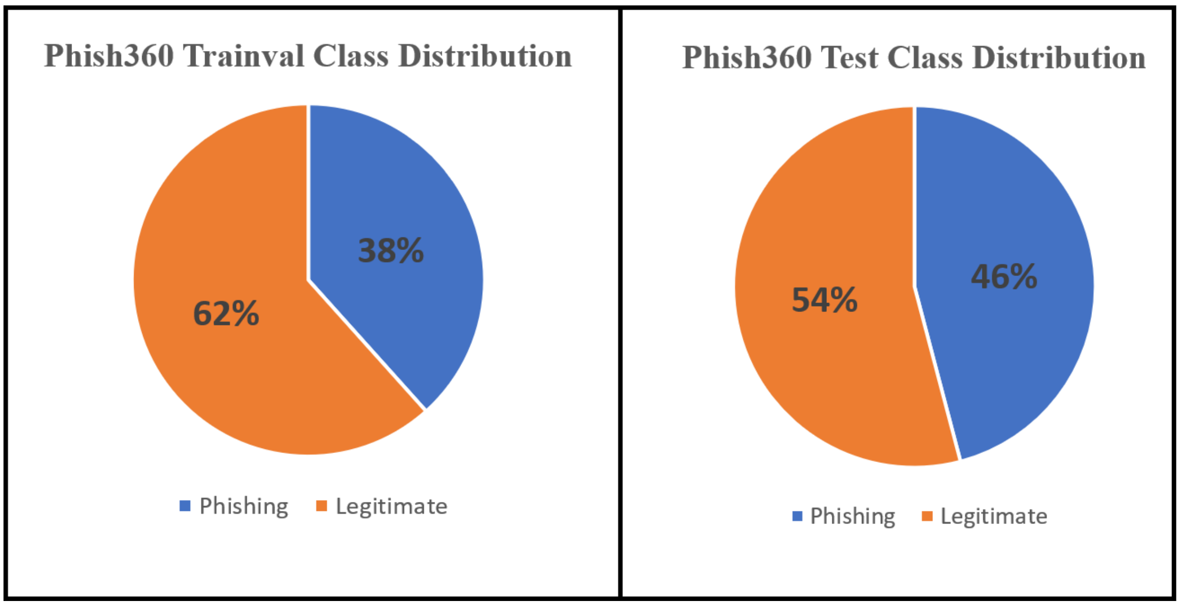


Figure 4.2 Phish360 class distribution in Trainval and Test folders

4.2.1. Phish360 Folder Structure

Every sample in the dataset contains four files in separate folders as depicted in Fig. 4.1, namely:

- **Label**: The label.txt file contains the targeted brand (e.g., Facebook) in phishing samples and 'legitimate' in legitimate samples.
- **RAW-HTML**: The index.html file contains the full HTML source code for the web page.
- **SCREEN-SHOT**: screen_shoot.png is a 1280x960 screenshot of the web page in .png format.
- **URL**: The url.txt file contains the full URL string of the web page.

In addition to the Label file, brand information of the phishing samples can also be accessed from the name of the folder sample, as the naming convention of phishing samples is in the form of (Pxxxx_brand). It is worth noting that brand information for test samples is

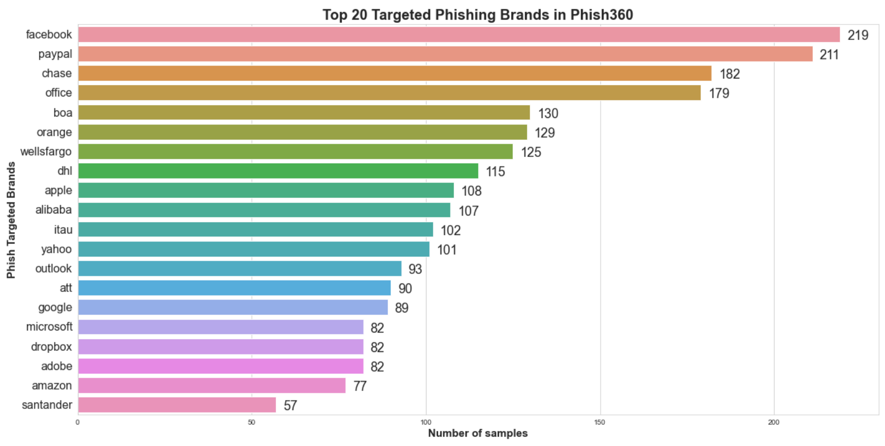


Figure 4.3 The top 20 phishing brands in Phish360 dataset

unavailable, hence recorded as ’unknown’. The top 20 phishing targeted brands and their counts are demonstrated in Fig. 4.3.

4.3. Benchmark Multimodal Datasets

As discussed earlier in 4.1., there is a lack of multimodal datasets in the Anti-phishing domain. As a result, researchers are compelled to collect their own data samples based on their specific needs. After a thorough review of the literature, we found four multimodal datasets that fit our criteria of including at least the main three raw data modalities (i.e., URL, HTML, and Screenshot Image), having diverse website samples from different sources and an adequate number of samples. In Figure 4.5 and 4.6, we highlight the folder naming convention and hierarchy of the benchmark datasets employed in this thesis, and Fig. 4.4 shows the sample size and class distribution of the benchmark datasets. Next, we provide an overview and explore the selected datasets in more detail.

4.3.1. PILWD-134K Dataset

The Phishing Index Phishing Index Login Websites Dataset (PILWD-134k) is a multimodal dataset consisting of 133,928 phishing and legitimate samples (class distribution shown in Fig. 4.4), collected and published by Sánchez-Paniagua et al. in [30]. The authors collected

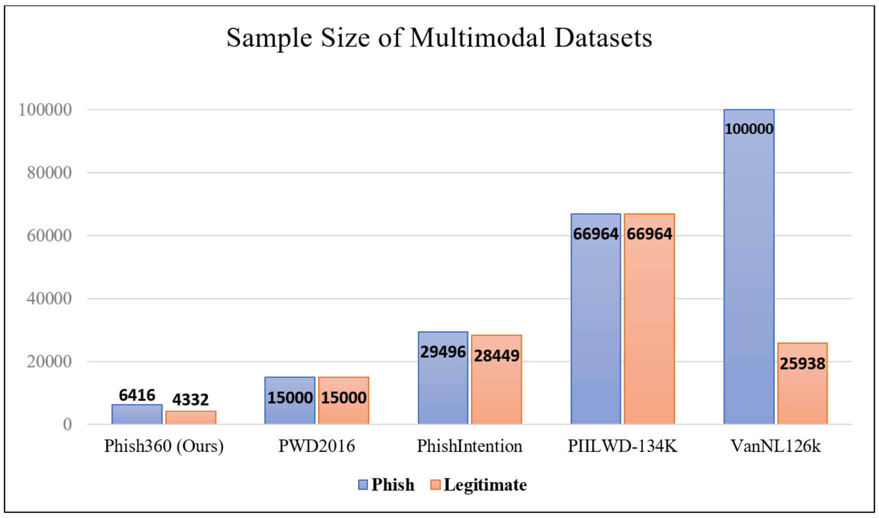
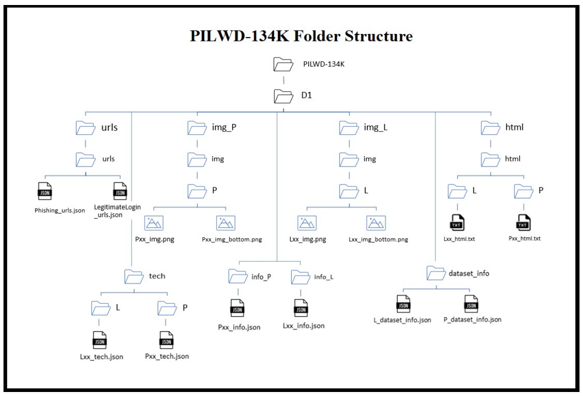
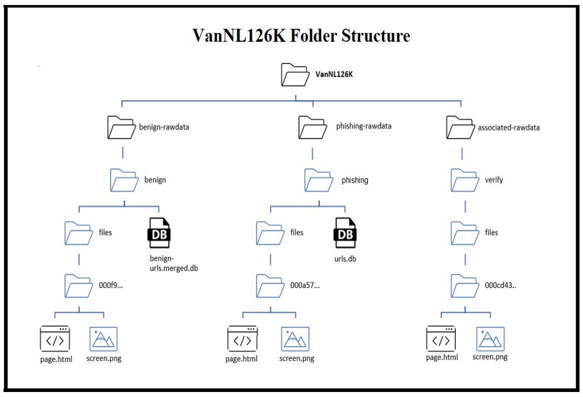


Figure 4.4 Multimodal datasets class sample distribution



(a)

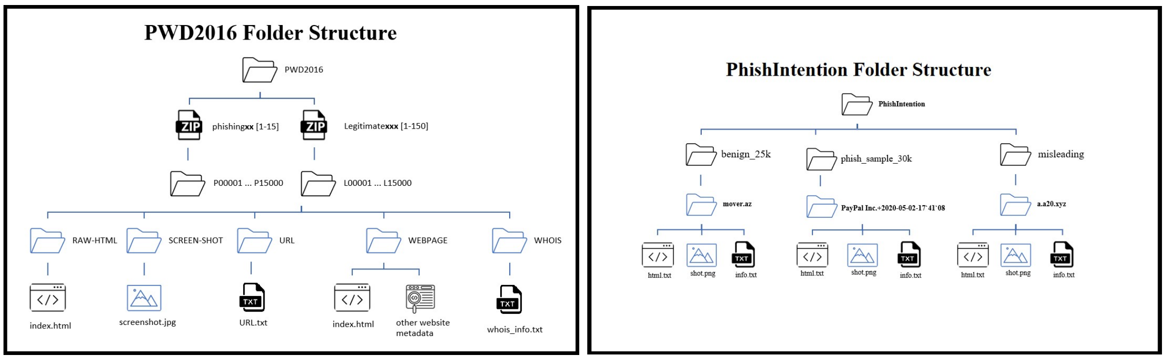


(b)

Figure 4.5 Folder hierarchy of the (a) PILWD-134K and (b) VanNL126k datasets

legitimate samples from Quantcast top sites and Majestic List, and phishing samples from PhishTank. The sample collection process was in 2019-2020, where for each sample in the dataset, the following raw data were collected:-

- URL
- HTML Source Code
- Two Screenshot Images (Web page's top and bottom)
- Additional data files like Technologies analysis, sample collection metadata, and WGET



(a) (b)

Figure 4.6 Folder hierarchy of (a) PWD2016 and (b) PhishIntention datasets

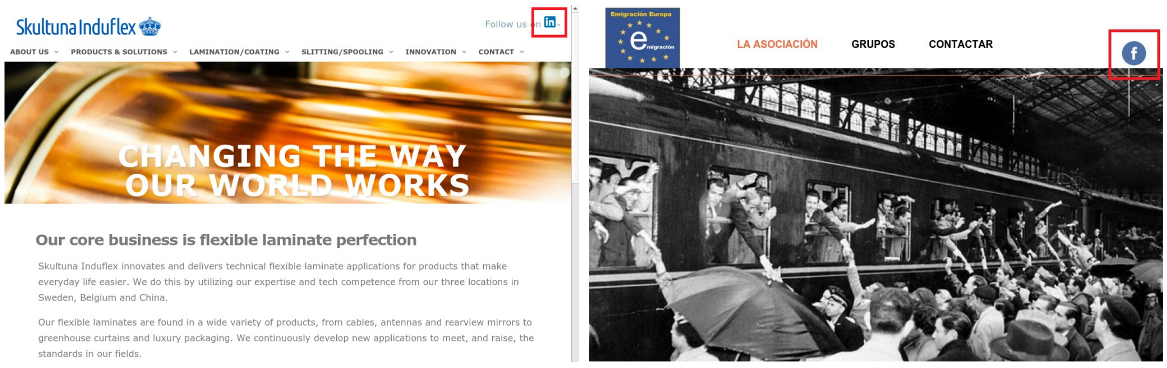
4.3.2. VanNL126K Dataset

The Phishing website dataset is a multimodal dataset consisting of 125,938 phishing and legitimate samples (see Fig. 4.4), collected and published by Van Dooremaal et al. in [31]. Throughout this thesis, we will refer to this dataset as *VanNL126k*. The authors collected legitimate samples from the DMOZ directory (to ensure the inclusion of less popular websites), and phishing samples from PhishTank, OpenPhish, and PhishStats in September-December 2019. The original dataset contains ”associated” legitimate samples that were obtained by a reverse search query. However, in this thesis, we use only phishing and legitimate samples. Each sample in the dataset has the following raw data:-

- URL
- HTML Source Code
- Screenshot Image
- Data Collection Metadata

4.3.3. PhishIntention Dataset

The PhishIntention dataset is a multimodal dataset published by Liu et al. [29], consisting of 29,496 phishing, 25,400 legitimate, and 3,049 misleading legitimate samples. The misleading legitimate samples are legitimate websites with links to social media websites



Our core business is flexible laminate perfection

Skultuna Induflex innovates and delivers technical flexible laminate applications for products that make everyday life easier. We do this by utilizing our expertise and tech competence from our three locations in Sweden, Belgium and China.

Our flexible laminates are found in a wide variety of products, from cables, antennas and rearview mirrors to greenhouse curtains and luxury packaging. We continuously develop new applications to meet, and raise, the standards in our fields.

(a) (b)

Figure 4.7 Misleading legitimate samples from PhishIntention dataset

like LinkedIn and Facebook or websites that allow sign-in and registration using Facebook and Google. We demonstrate two misleading legitimate samples in Fig. 4.7 highlighting the social media logos by red rectangles. In this thesis, we combine both legitimate and misleading legitimate samples into the legitimate class (see Fig.4.4). The authors collected legitimate samples from Alexa and phishing samples from OpenPhish in 2019-2020. Each sample in the dataset has the following raw data:-

- URL
- HTML Source Code
- Screenshot Image
- Additional data files like coordinates of some objects in the Image

4.3.4. PWD2016 Dataset

The Phishing Websites Dataset (PWD2016) is a multimodal dataset published by Chiew et al. in [32], consisting of 30,000 phishing and legitimate samples collected in 2016 (class distribution shown in Fig. 4.4). The phishing samples were collected from PhishTank, and to ensure the inclusion of less popular legitimate websites in different languages, the authors collected legitimate samples from the DMOZ directory, BOTW, and Alexa. Each sample in the dataset has the following raw data:-

- URL
- HTML Source Code
- Screenshot Image
- Additional data files like favicons, JavaScript files, and WHOIS information

4.4. Data Processing: Cleaning, Extraction, and Representation

In this subsection, we provide an overview of the datasets' pre-processing stages by detailing how we extracted the data modalities, as well as our choice for data representation. Figure 4.8 abstractly demonstrates the overall procedure. As shown in Fig. 4.8, we handle the different data sources from the multimodal datasets in a column-oriented fashion, storing the information in a Panda's DataFrame, which makes column retrieval easier. As storing all the multimodal datasets in a single DataFrame is memory-inefficient, we opted for saving each dataset into a separate DataFrame. For each multimodal dataset, we applied the same Python procedure shown in Fig. 4.8 to create the DataFrames. We also added additional important columns to the DataFrames, which are detailed in the following subsections.

To locally store the produced DataFrames using an efficient file format, we've considered multiple file format options. Contemplating about (1) The large number of DataFrame columns and rows, (2) Textual information loss and the large size of long textual data (i.e., HTML code and extracted texts), (3) Disk space requirements and frequent reading operations of specific columns (detailed in later sections), and most importantly, (4) The compatibility of the file format with different programming languages as we are planning to share the data with the research community. Considering the mentioned restrictions, we chose the *Parquet* file format, which is a column-oriented format designed for efficient data storage and retrieval. The parquet format is designed to handle all Pandas data types while providing an efficient columnar binary serialization for DataFrames. Its column-oriented nature makes retrieving single or multiple columns possible in a space and time-efficient manner while being compatible with multiple programming languages (e.g., Java, C++, and Python) [87]. In the code snippet below, we demonstrate how we can fetch selected columns

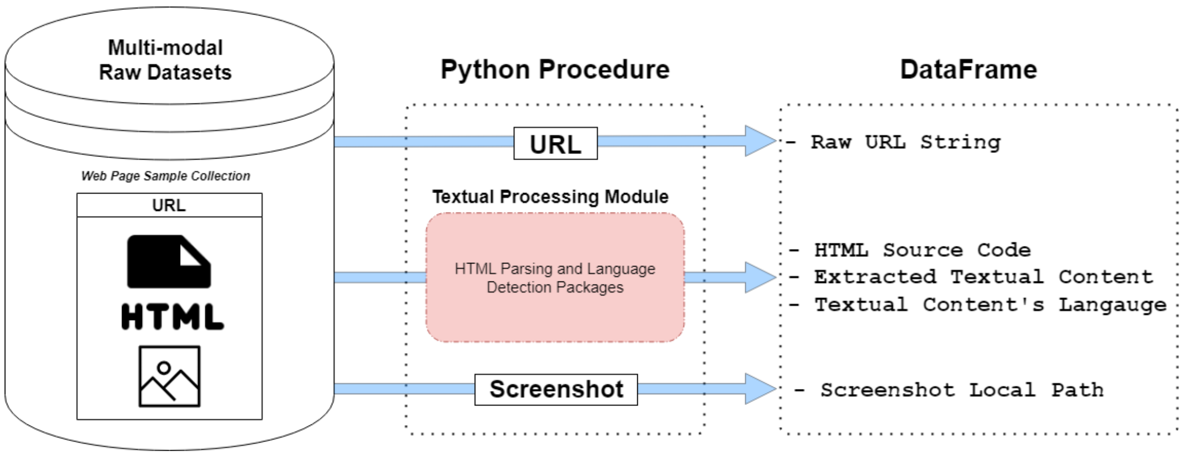

Multi-modal
Raw Datasets
Web Page Sample Collection
URL
HTML
Python Procedure
URL
Textual Processing Module
HTML Parsing and Language Detection Packages
Screenshot
DataFrame
- Raw URL String
- HTML Source Code
- Extracted Textual Content
- Textual Content's Langauge
- Screenshot Local Path

Figure 4.8 This figure illustrates the Python procedure implemented for Multimodal raw data processing into a DataFrame

rather than reading the whole parquet file with the Pandas package in Python. Thus providing an efficient method to fetch only the needed columns into memory. This is crucial because we are dealing with multiple datasets having a large number of textual observations (rows).

```
df = pandas.read_parquet('file_name.parquet', columns = ['column_name_1', 'column_name_2'])
```

Unlike the other explored alternatives (e.g., CSVs, pickle, SQL, and feather files), the option to fetch selective columns with no limitation for the long text is only available in parquet files.

4.4.1. DataFrame Creation Procedure

In this step, after obtaining all the samples from the different multimodal datasets employed, we apply the procedure depicted in Fig. 4.8. Considering that there are minor differences in the procedure when applied to the different datasets as their folder hierarchy slightly differs (see Fig.4.5 and Fig. 4.6).

As mentioned before, our dataset contains three main data modalities: URLs, HTML source code, and a screenshot of the top part of the webpage. Unlike other public multi-modal datasets, we double-checked the validity of the samples after filtering, making sure that every sample in the dataset contains these three data modalities. We demonstrate the employed approach to represent the DataFrame in *Algorithm 1.* Since the employed benchmark datasets have missing and corrupt file samples, we had to implement a workaround to account for this issue.

Algorithm 1 Python procedure to generate the DataFrame from Raw data with Phish360 dataset

```
0:  for each folder_name in directory do
1:      begin
2:      Create a dictionary where key = folder_name.
3:      Initialize the list of values for folder_name with datasetname ('Phish360').
4:      Append folder_name to the list.
5:      Identify and append the sample's class ('legit' or 'phish' using folder_name or label.txt).
6:      Determine the paths of URL, HTML, and Image files of the sample.
7:      if url.txt exists and URL string read is successful then
8:          Append the URL string to the list.
9:      else
10:         Append None to the list.
11:     end if
12:     if screen_shoot.png file exists then
13:         Append the full path to the list.
14:     else
15:         Append None to the list.
16:     end if
17:     if index.html exists and reading it is successful then
18:         Append the HTML text string to the list.
19:         Extract and append text from HTML string using different parsing tools.
20:     else
21:         Append None for the HTML and extracted text strings.
22:     end if
23: end for
24: return dictionary
```

Next, we needed to extract the plain text from the website's HTML source code using HTML parsing tools (Textual Processing Module in Fig. 4.8). Using the algorithmic approach demonstrated in *Algorithm 1*, we utilize five different HTML parsing packages available in Python to extract plain text from HTML:

1. **Trafilatura** [88]
2. **BeautifulSoup** [89]
3. **html2text** [90]
4. **lxml** [91]
5. **html_text** [92]

The packages include methods that accept the HTML code string as input and return the plain extract text as output. All the methods employed were used with default parameters. It is worth noting that the length, spacing, and text structure of the extracted plain text varies between the employed parsing tools. Table 4.1 demonstrates the resultant DataFrame by providing a description of each column in a generic fashion.

Column Name	**Description**
dataset_name	Name of the dataset
folder_name	Name of folder/file in the dataset
Class	The Class of the sample (phish: 1/legit: 0)
URL	The full URL of the sample
image_path	The relative local path of the screenshot Image
trafilatura_text	Extracted textual content using *trafilatura* package
trafilatura_text_language	The language detected from extracted plain text
beautifulSoup_text	Extracted textual content using *BeautifulSoup* package
beautifulSoup_text_language	The language detected from extracted plain text
html2text_text	Extracted textual content using *html2text* package
lxml_text	Extracted textual content using *lxml* package
html_extract_text	Extracted textual content using *html_extract* package
full_html	Web page's Source Code

Table 4.1 A description of the columns in the created DataFrame containing the extracted features.

In addition to the features shown in Table 4.1, we extract several additional features that are imperative to investigate dataset sample quality and diversity (e.g., top-level domain, domain, and targeted brand). Moreover, these features are essential for an Exploratory Data Analysis (EDA) of these multimodal datasets and are useful for the research community. Finally, since the screenshot and URL will be used as raw input, we stored the URL string and the local path of the screenshot in the URL and image_path columns, respectively. To highlight the difference between the extracted text using the text extraction methods and the fields of each column, we show a random phishing sample from our Phish360 dataset in Table 4.2.

After processing every multimodal dataset separately by creating a single DataFrame for every dataset (including our dataset Phish360). We locally stored each dataset's DataFrame into two separate Parquet files based on the samples' class (Phish360_legit.parquet and

Column Name	Value
dataset_name	Phish360
folder_name	P11258
class	phish
brand	adobe
URL	http://jaywatsonfiles.000webhostapp.com/
TLD	com
Domain	000webhostapp
FLD	000webhostapp.com
Subdomain	jaywatsonfiles
SSL	False
image_path	phish360/trainval/P11258_adobe/SCREEN-SHOT/screen_shoot.png
trafilatura_text	Adobe PDF Online/nAccount/nSign In/nConfirm yo...
trafilatura_text_language	English
beautifulSoup_text	/n/n/nAdobe PDF/n/n/n/n/n/n/nAdobe PDF Onlin...
beautifulSoup_text_language	English
html2text_text	| Adobe PDF Online | | Account | Sign I.../
lxml_text	/n/n/nAdobe PDF/n/n/na/n color:#454444;/n t...
html_extract_text	Adobe PDF/n/nAdobe PDF Online Account Sign In/...
full_html	<html dir="LTR" lang="en"><head>/n<meta http-...
html_text_language	English

Table 4.2 A sample from Phish360 dataset showing the possible values for the different columns in the DataFrame, highlighting the difference between extracted text using different parser tools.

Phish360_phish.parquet). As we have five total datasets (phish360 + 4 benchmark datasets), a total of 10 parquet files were created. In the Next subsection, we utilize these parquet files to further inspect the multimodal datasets.

4.5. Multimodal Datasets Analysis

In this step, we first analyze the multimodal datasets by inspecting the sample files, followed by a URL analysis to investigate the diversity of the samples in each dataset. Finally, we dive deeper into the remaining data modalities by investigating the processed data using the created parquet files.

4.5.1. Missing Data Files

One of the most common issues with datasets is missing data samples. Hence, we analyzed the samples to check for missing and corrupt files within the multimodal datasets. The number of missing samples in each class is presented in table 4.3. Unlike the benchmark datasets, our collected dataset, *Phish360*, has no missing or invalid files. The numbers

Dataset Name	**Sample Size**		**URL**		**HTML**		**Screenshot**	
	Phish	Legit	Phish	Legit	Phish	Legit	Phish	Legit
PWD2016	15,000	15,000	2	1	1,315	363	588	398
PhishIntention	29,496	28,449	0	3,426*	450	6,991*	0	0
PILWD-134K	66,964	66,964	0	0	0	31	0	11
VanNL126K	100,000	25,938	0	0	0	3,655	71	3,659
Phish360 (OURS)	**6,416**	**4,332**	**0**	**0**	**0**	**0**	**0**	**0**

*All Missing files from benign category (no missing files in misleading samples)

Table 4.3 Number of missing/corrupt sample files in multi-modal datasets.

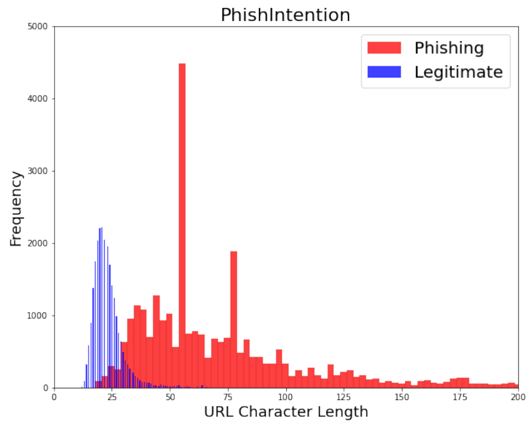
representing the missing samples in table 4.3 are obtained by first checking the existence of the files inside the sample folders. Then, we added the number of unreadable corrupt files due to several errors, e.g., file naming, invalid file extensions, or encoding errors in HTML files.

4.5.2. URL Analysis

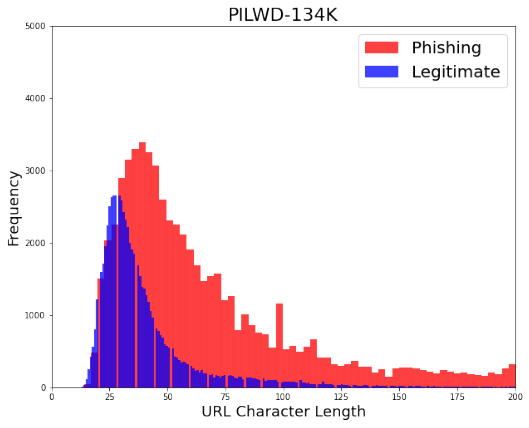
In this step, we aim to uncover trends and hidden patterns in the URL distribution by exploring the differences between legitimate and phishing URLs in the datasets. Even though our proposed scheme utilizes automatic feature extraction, which demonstrates superiority over manual feature extraction methods in the literature. However, a concise EDA exploring the trends among the employed datasets will be beneficial to the research community in helping them choose the appropriate dataset that fits their research needs.

We first start with plotting URL length histograms to view the overall distribution of the data as well as the frequency of every value, demonstrating the most and least common values. Upon examining figure 4.9, we immediately notice the distinct differences in the URL length distribution across the classes and between the datasets. Despite URL samples exceeding 200 characters, we limited the x-axis and y-axis range to 200 and 5,000, respectively, to ensure consistency and comparability and avoid any potential bias in observations.

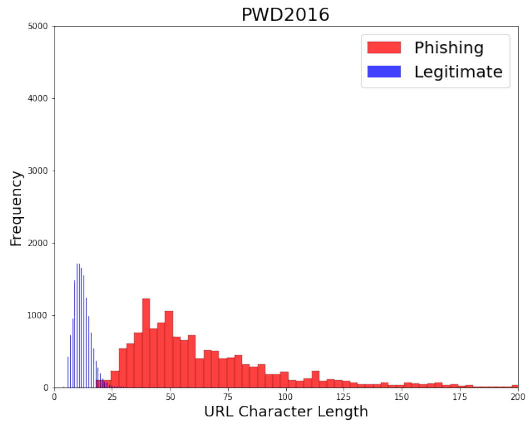
The most apparent URL length difference between phishing and legitimate samples is observed in PWD2016 (sub-figure c in 4.9), where legitimate URLs are considerably shorter



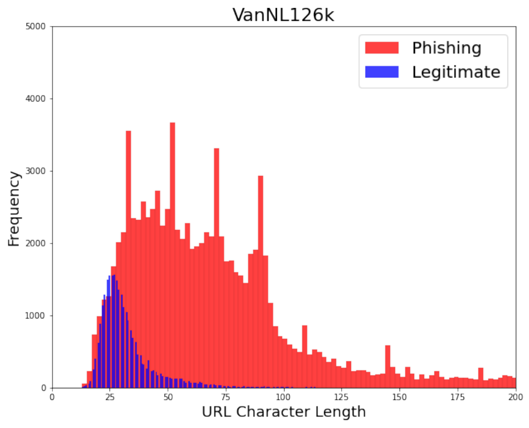
(a)



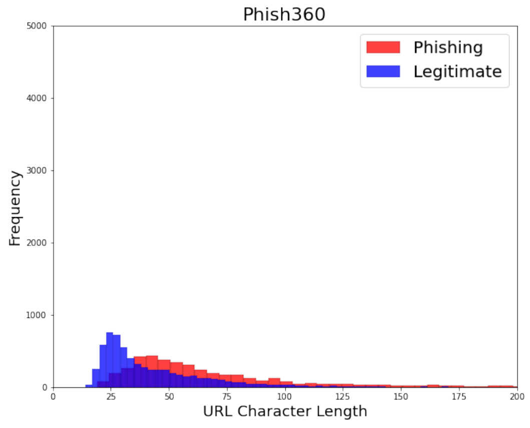
(b)



(c)



(d)



(e)

Figure 4.9 Histograms showing the distribution of URL character lengths for benchmark datasets: (a) PhishIntention, (b) PILWD-134K, (c) PWD2016, (d) VanNL126k, and (e) Phish360. The maximum values for the x and y axes were set to 200 and 5,000.

compared to phishing ones. After examining the URL samples in PWD2016, we noticed that legitimate URLs in the datasets lack most URL components and solely consist of the domain and top-level domain (e.g., facebook.com, amazon.uk, etc.). This is one of the critical issues in PWD2016, which potentially leads to biased results for URL-based approaches. Similarly, most legitimate URL samples in PhishIntention and VanNL126k datasets (sub-figures a and d in 4.9) are shorter than phishing ones, where the highest frequency is around 20-30 characters. Since phishers attempt to direct victims to a malicious website containing input fields rather than homepages to steal their information, they use the URL path and query parameters to direct victims to the specific landing page, increasing the URL's character length.

Therefore, anti-phishing datasets must incorporate legitimate login web pages instead of solely focusing on home pages. Unlike home pages, login web pages feature login forms used to authenticate users, resulting in URL lengths similar to phishing URLs. Similarly, phishing sites incorporate these login forms to obtain and steal the victim's credentials. Some anti-phishing solutions like PhishWHO [79] consider suspicious pages legitimate since they are harmless and do not feature an input form. That is why it is imperative to include legitimate login web pages.

Examining the length distribution of legitimate URLs in our dataset, Phish360, and PILWD-134K (sub-figures e and b in 4.9), it is evident that the inclusion of legitimate login web pages increases the overall length of legitimate URLs. This makes the differentiation between phishing and legitimate URLs based on the length of the URL more challenging for detection systems and helps to avoid bias, especially in rule-based and hand-crafted machine learning models. Additionally, the URLs of legitimate login web pages boost the overall model generalization ability, especially in real-world scenarios [30]. That is because current anti-phishing works are shifting towards representation learning approaches utilizing automatic feature extraction methods, which dictate a standard distribution of the URL character length [85].

Since there is no widely accepted method to analyze the diversity of anti-phishing datasets,

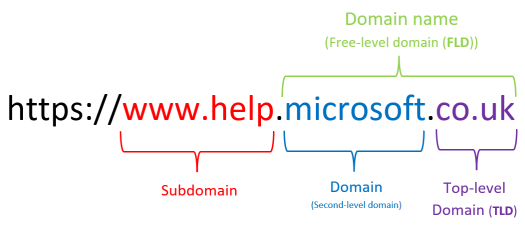
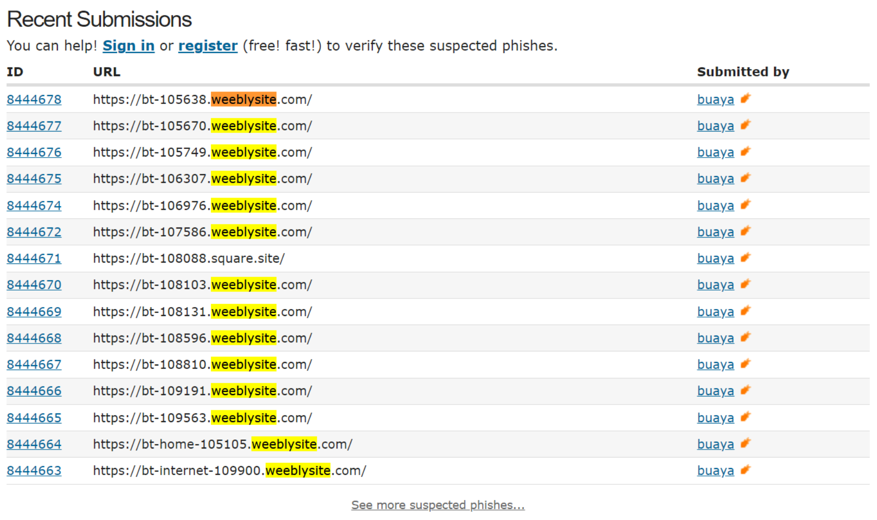


Figure 4.10 URL example demonstrating the different components used to analyze sample diversity


Recent Submissions

You can help! **Sign in** or **register** (free! fast!) to verify these suspected phishes.

ID	URL	Submitted by
8444678	https://bt-105638.weeblysite.com/	buaya
8444677	https://bt-105670.weeblysite.com/	buaya
8444676	https://bt-105749.weeblysite.com/	buaya
8444675	https://bt-106307.weeblysite.com/	buaya
8444674	https://bt-106976.weeblysite.com/	buaya
8444672	https://bt-107586.weeblysite.com/	buaya
8444671	https://bt-108088.square.site/	buaya
8444670	https://bt-108103.weeblysite.com/	buaya
8444669	https://bt-108131.weeblysite.com/	buaya
8444668	https://bt-108596.weeblysite.com/	buaya
8444667	https://bt-108810.weeblysite.com/	buaya
8444666	https://bt-109191.weeblysite.com/	buaya
8444665	https://bt-109563.weeblysite.com/	buaya
8444664	https://bt-home-105105.weeblysite.com/	buaya
8444663	https://bt-internet-109900.weeblysite.com/	buaya

See more suspected phishes...

Figure 4.11 Recent PhishTank URL submissions showing near-duplicate phishing URLs that use the same domain.

some studies have used the number of unique domains and top-level domains (TLD) to measure it. To investigate the diversity of the samples, we computed the unique percentages of URL samples, domains, TLD, free-level domain (FLD, also known as Domain Name), and subdomains. Figure 4.10 demonstrates these URL components. The diversity metrics for phishing URL samples are presented in table 4.5, and the legitimate ones in table 4.4. Comparing the percentages in the legitimate and phishing tables, it is evident that legitimate samples in all datasets are far more diverse than phishing samples. That is because there are far more legitimate websites on the internet, and obtaining legitimate URL samples from online repositories and ranking lists is generally easier.

However, in the case of phishing URLs acquired from repositories such as PhishTank, it is common to have consecutive submissions originating from the same domain but with various subdomains, as demonstrated in figure 4.11. This pattern indicates that attackers often create

Dataset Name	Legit Domain Stats				
	Unique URLs (%)	Unique Domains (%)	Unique TLD (%)	Unique FLD (%)	Unique Subdomains (%)
PWD2016	**100.0**	**92.97**	2.46	-	0.65
PhishIntention	87.94	82.98	2.17	86.68	3.21
PILWD-134K	99.23	91.37	0.76	**92.57**	3.27
VanNL126K	**100.0**	84.90	2.16	85.68	**9.94**
Phish360 (OURS)	99.41	88.73	**3.07**	88.92	7.29

Table 4.4 Comparison of the percentage of unique legitimate URL domains among the multimodal datasets.

Dataset Name	Phish Domain Stats				
	Unique URLs (%)	Unique Domains (%)	Unique TLD (%)	Unique FLD (%)	Unique Subdomains (%)
PWD2016	38.25	17.71	1.40	17.85	2.74
PhishIntention	87.21	42.62	1.56	43.04	24.46
PILWD-134K	86.68	45.23	1.08	46.41	21.35
VanNL126K	**100.0**	25.91	0.67	26.85	13.65
Phish360 (OURS)	98.26	**73.63**	**6.69**	**73.86**	**28.69**

Table 4.5 Comparison of the percentage of unique phishing URL domains among the multimodal datasets.

multiple subdomains under the same domain, using redirections to prolong the lifespan of their phishing sites. To address this issue, in our sample collection process of Phish360, we have minimized the near-duplicate URL samples by spacing out the collection process over extended time periods. Compared with the benchmark datasets, our approach ensures a more diverse range of phishing samples, as evidenced by the stats in table 4.5.

Upon examining the percentage of unique phishing URLs in table 4.5, we observe an extremely low level of uniqueness in PWD2016 URL samples. The high number of exact duplicate URL samples in PWD2016 poses a critical issue, as it leads to the well-known problem of data leakage in machine learning. Data leakage occurs when testing samples are 'leaked' into the training portion during model training. One cause for the data leakage is duplicate samples, leading to overoptimistic models achieving high results during the model development phase. However, the model underperforms and achieves poor results when tested on new data. Therefore, datasets should mostly contain unique data samples to prevent overlapping train and test samples. This issue is worth highlighting because a single-modal URL-based approach would suffer from data leakage if duplicate URL samples are not minimized, which leads to biased results and decreases the model's generalization capability.

4.5.3. HTML and Image Analysis

As previously mentioned, the datasets contain numerous instances of repeated and duplicate URL samples. However, relying solely on the count of unique URLs is insufficient to determine whether a sample is repeated. Therefore, it is essential to examine the other data sources as well. We start by analyzing the HTML source code and the extracted text using the employed parsers. After thoroughly examining the extracted text using the parsing tools, it became evident that the extracted plain text using html2text, lxml, and html_text contains many HTML code segments, symbols, and insignificant garbage values. This observation suggests that the extracted text using Trafilatura (TF) and BeautifulSoup (BS) would provide more informative textual content. Upon inspecting the texts extracted using TF and BS, we selected the extracted text using the BS package, as it provides more comprehensive textual content, including essential details such as the web page title.

In figure 4.12, we compare the datasets by computing the percentage of unique HTML and BS text for both phishing and legitimate classes. As anticipated, the percentage of unique HTML phishing samples is significantly lower than that of the URLS. This discrepancy is because phishers often reuse the same HTML source code with different URLs when the site gets detected and shut down. However, due to manual sample verification during the collection process and the diversification of phishing sample sources, we observe a notably high percentage of 96.63 unique HTML phishing samples in Phish360. Additionally, spreading out the sample collection time ensures the uniqueness and diversity of the website samples, encompassing a wide range of phishing tactics.

A significant limitation of single-modality in anti-phishing is the tendency for data leakage due to the high percentage of duplicate samples within anti-phishing datasets. Therefore, one of the benefits of utilizing multimodal datasets is the reliance on multiple data sources. The multimodality ensures the uniqueness of training samples, even when the HTML or URL is an exact or near-duplicate. An informative approach to evaluate the uniqueness of multimodal samples is to consider each web page sample as a triplet entity composed of the three primary data modalities: URL, HTML, and screenshot image. We explored

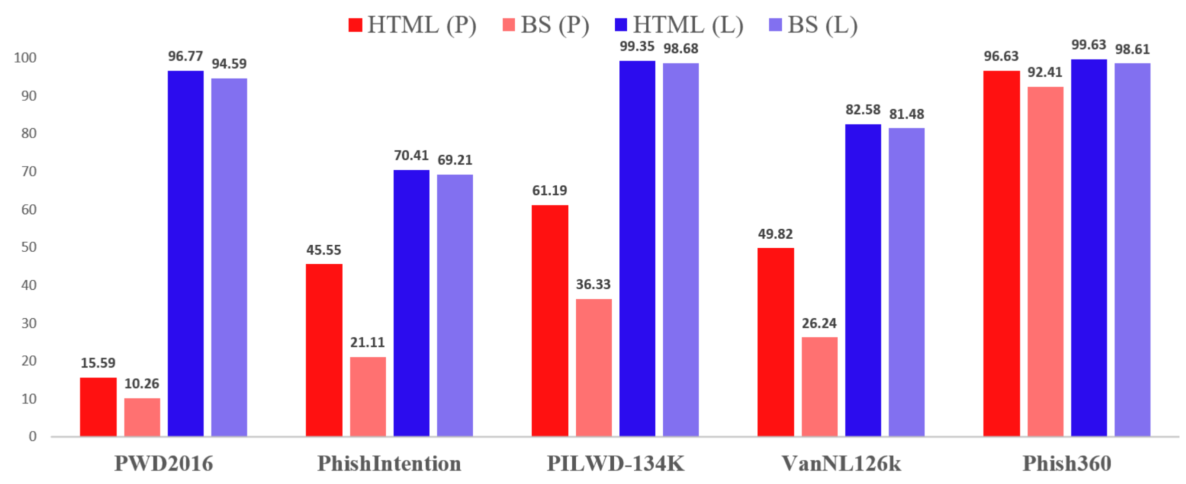
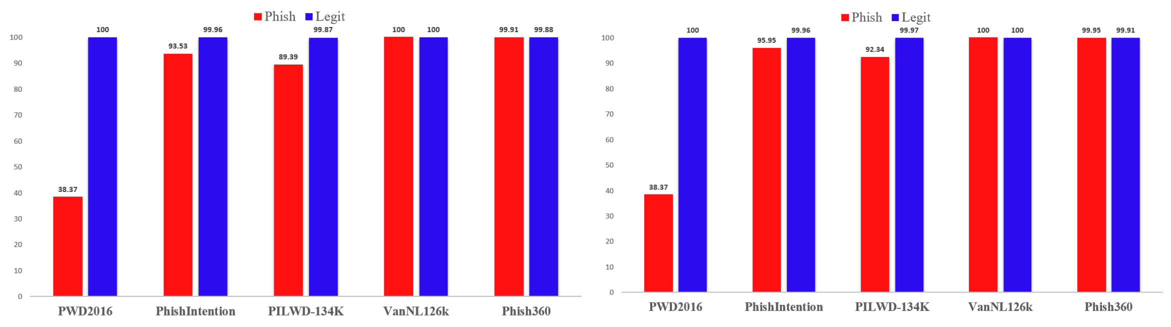


Figure 4.12 Comparison of the percentages of unique HTML and BeautifulSoup text (BS) for phishing (P) and legitimate (L) classes across multimodal datasets.

the uniqueness of multimodal data samples by creating a set for each sample containing the URL and HTML strings. For the screenshot image representation, we had to devise a method to generate a unique representation for each image sample. Thought this task could be achieved by utilizing an image model such as ResNet or DenseNet to produce a unique feature vector. However, given the large number of image samples in all the datasets, we opted for hash functions to compute a hash value for every image in the datasets, which demands less computational resources. To accomplish that, we chose the famous SHA-256 cryptographic hash algorithm that produces a unique fixed-size output (hash value) for each unique input image.

The resulting unique identifier for a web page sample comprises a set containing: (URL, HTML, hash(image)), as illustrated in figure 4.13. Comparing the percentage of unique single-modal samples (figure 4.12) with multimodal triplet samples in figure 4.13, we observe a substantial increase in the percentage of unique samples. The multimodal representation of phishing samples mitigates the data leakage tendency within the anti-phishing datasets. Therefore, a side benefit of multimodality is the reliance on multiple data sources, which makes the data samples unique even if the HTML or URL is an exact or near-duplicate. Additionally, since we intend to employ the extracted plain text rather than the HTML source code, we also examined the uniqueness of the data by



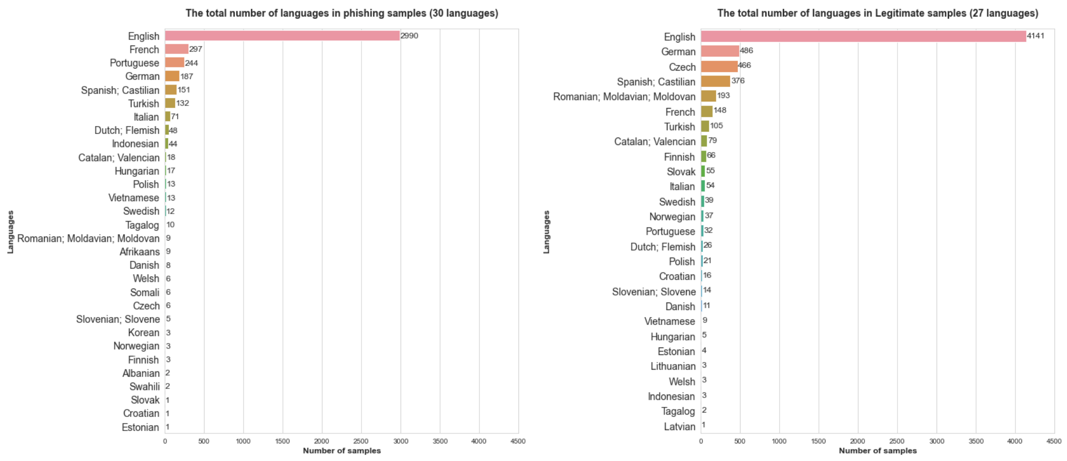
(a) (b)

Figure 4.13 Comparison of the percentage of unique triplet representation of samples (a) (URL, BS_text, image_hash) (b) (URL, HTML, image_hash) across multimodal datasets.

representing it as triplets of URL, extracted textual content (BS or TF), and a hash value of the image, as presented in figure 4.13. Notably, the percentages demonstrated in subfigures a and b 4.13 are calculated by dividing the number of unique triplets by the number of valid samples. The number of valid samples refers to the samples having a valid URL, HTML, and image; samples missing one of these data sources are eliminated.

An important diversity measure often overlooked within anti-phishing datasets is the linguistic composition of the website samples. Diversifying the languages included in the datasets is essential given the multilingual nature of phishing websites, which target more than just English-speaking users. Recent APWG phishing activity trend reports have shown a significant increase in attacks across various countries. Therefore, datasets must include a wide range of different languages. To this end, we incorporated several widely spoken world languages, such as English, German, French, Spanish, and Portuguese. In our Phish360 collection process, we ensured the diversity of the samples by including 30 languages for phishing samples and 27 for legitimate ones. Employing the *langdetect* Python package for language detection, we identified the languages using extracted plain text, and we present the language distribution of Phish360 samples for both classes in figure 4.14. Similarly, we highlight the language distribution within the benchmark datasets, illustrating the number of distinct languages in figure 4.15.

A widely utilized metric for assessing the website categories is the legitimate targeted brands



(a) (b)

Figure 4.14 The distribution of website languages in Phish360 in (a) phishing and (b) legitimate samples

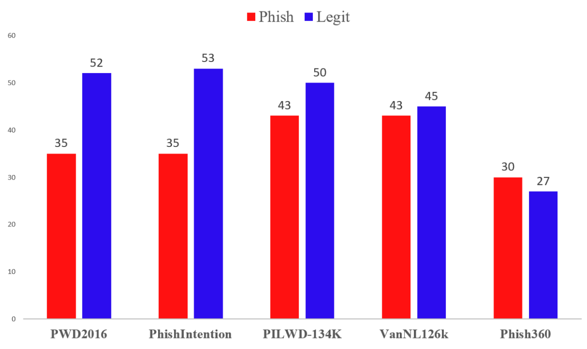


Figure 4.15 The language distribution within phishing and legitimate website samples across multimodal datasets.

among the phishing samples. Analyzing phishing samples to explore the impersonated brands provides valuable insight into the categories of the samples. While certain well-known brands, especially in financial and payment services like PayPal and American Express, are targeted more frequently, it is crucial to incorporate a wide range of website categories, including government/institutional, social media, and E-Commerce/retail. Figure 4.16 presents the top 20 targeted brands among the datasets, with brand information available in all datasets except PWD2016, which is regarded as 'unknown'. Approximately 36% and 88% of targeted brands are considered unknown in PILWD-134K and VanNL126k, respectively (subfigures b and d in 4.16). The 'phish' brand category in Phish360 (subfigure c 4.16) refers to phishing test samples lacking brand information.

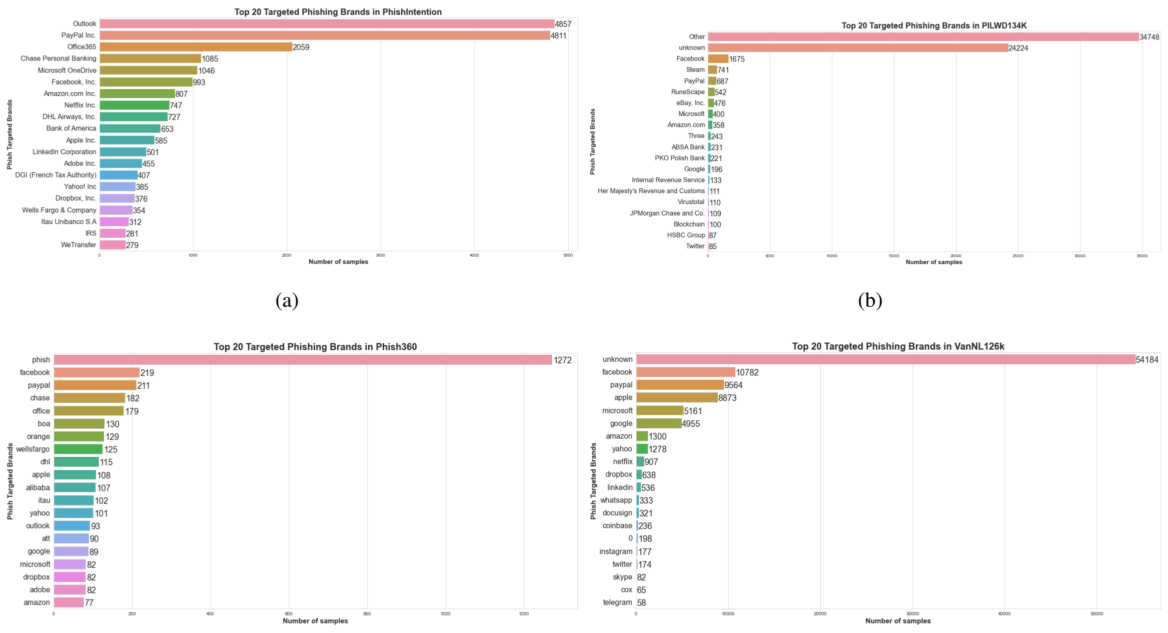


Figure 4.16 Top 20 targeted legitimate brands within the multimodal datasets: (a) PhishIntention, (b) PILWD134K, (c) Phish360, and (d) VanNL126k.

Screenshot Image Analysis: Upon examining the screenshot images within the benchmark datasets, we observed a discrepancy in their resolution dimensions. Rather than having a consistent image resolution across all the screenshots, we noted variations in their dimensions. The image dimensions (expressed as width*height) for all the datasets are as follows:

- **PhishIntention:** 38% of the screenshot image sizes are 1920*1080, 19% are 1366*768, and the remaining images are spread across 7,897 different dimensions.
- **PWD2016:** 6% of the screenshot image sizes are 510*1330, 2% are 18*18, and the remaining images are spread across 14,836 different dimensions.
- **PILWD-134K:** 68% of the screenshot image sizes are 1906*922, 22% are 1853*922, and the remaining images are spread across 26 different dimensions.
- **Phish360:** 99.8% of the screenshot images are 1280*960 and the remaining images are spread across 8 different dimensions.

- **VanNL126k:** All screenshot images are 1280*768.

5. THE PROPOSED APPROACH

5.1. Motivation

Reviewing the existing anti-phishing studies in the literature, we reveal several limitations that attackers can exploit. We summarize and highlight the drawbacks of existing approaches in the anti-phishing domain and define our motivation. A critical issue in the anti-phishing research field is the lack of standard benchmarking datasets; this entails several other issues that we previously discussed in the DATASETS section. Given the lack of a standardized anti-phishing dataset, researchers are compelled to collect their own datasets based on their specific tasks and needs by adopting certain criteria and standards.

The first drawback of the existing anti-phishing approaches is applying manual feature engineering (extracting hand-crafted features). Although manual feature engineering has several advantages in different domains, it is complicated and inefficient in the anti-phishing domain. That is because it requires extensive technical skills and domain expertise. Additionally, it is a time-consuming task that may only capture some relevant information from the data, making it ineffective and impractical against the highly evolving and volatile state of phishing tactics. Similarly, language-dependent detection systems (usually English, since most sites are English websites) fail to detect phishing websites prepared in a non-English language. The inherent nature of such detection systems makes it almost impossible to detect zero-day attacks. Although some hand-crafted features are discriminative, the robustness of the selected features is not examined, enabling easy manipulation by phishers. For example, past works incorporated the existence of HTTPS protocol in URLs and the length of the URL as highly discriminative features. These features, however, became impractical as phishers easily manipulated them. In other words, manual features are vulnerable once they have been discovered.

The second main limitation is third-party dependency, where an approach or an extracted feature depends on a third-party service. For example, it is a common habit in image-based approaches to use the logo, favicon, or regions of the image in a reverse image search using search engines, directly affecting the approach's overall accuracy. Search engine-based approaches usually suffer a high false positive rate (misclassifying legitimate websites as phishing) because newly created legitimate or unfamous websites are not highly indexed on search engines like Google, Bing, or Yandex [93]. Additionally, identifying phishing landing pages hosted on compromised legitimate websites or legitimate websites that are employed to obtain sensitive information maliciously (e.g., Google Forums) would be extremely challenging [93]. Moreover, third-party dependent features like domain age, WHOIS information, and ranking lists create a latency issue that decreases the system's runtime performance and robustness.

The third limitation to be aware of is the use of single-modality, which refers to detection systems that rely on a single information source. Such detection systems are often easy for attackers to bypass by exploiting their vulnerabilities. As previously mentioned, attackers use various obfuscation and evasion tactics to evade these systems, including replacing text with images, cloning the HTML code structure, HTML and Javascript obfuscations, and link shortening or redirections. Furthermore, there is a lack of utilization of multi-modality despite the demonstrated superiority of combining multiple data sources (bimodal/hybrid approaches) over single-modal approaches in the literature.

The last and major issue which is from a research perspective, is that most anti-phishing studies do not provide explicit instructions or steps to reproduce their experimental findings. Essential details to obtain reproducible results include the train-test split ratio and seed value, employed dataset details and class distribution, and model hyperparameters. *Computational reproducibility* [94] refers to the ability to recreate models and results of past work straightforwardly. The reproducibility is imperative as it increases the confidence in produced results and expands the boundaries of discoveries, enabling scientific progress [94]. Therefore, publishing the implemented codebase allows model replication, evaluation experiments with new datasets, and fair comparison analysis with other detection systems.

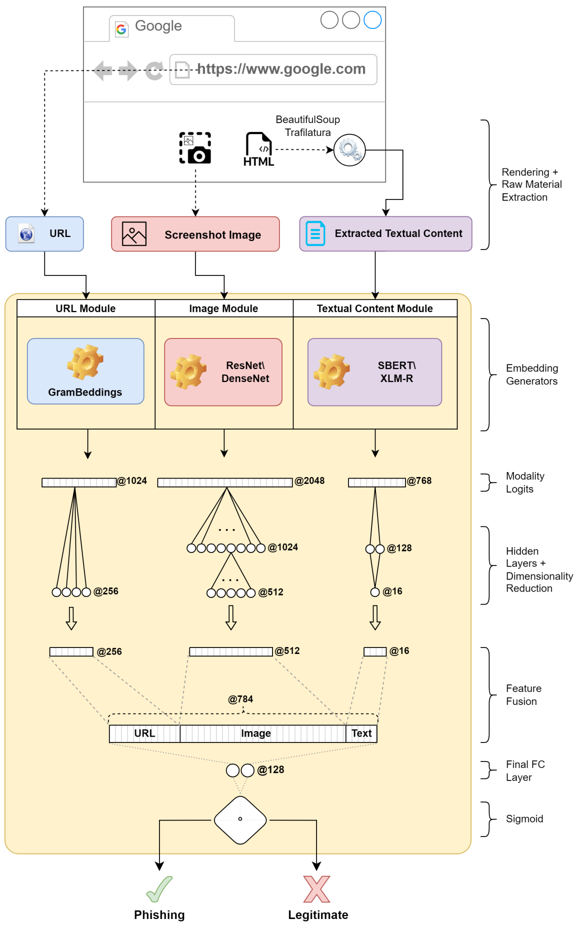


Figure 5.1 *CrossPhire*: Overview of our proposed deep neural network architecture

5.2. CrossPhire

Our approach *CrossPhire* is an end-to-end deep learning model that identifies phishing samples using website raw multimodal data sources with minimal preprocessing. In contrast to traditional phishing detection approaches, CrossPhire leverages automatic feature extraction using the web page's raw URL, screenshot image, and extracted textual content. The website's textual content is extracted from the HTML source code using HTML parsers as a preprocessing step, while the website's URL and the screenshot image are utilized without any preprocessing. Notably, utilizing the website's extracted markup-free textual content has never been employed as an information source in the phishing website detection domain.

We hypothesize that the textual content of a website contains valuable information that can effectively distinguish between phishing and legitimate websites. Unlike existing phishing detection approaches that rely on Model Stacking, our framework is a novel multimodal architecture that operates on distinct raw data modalities by combining them, as illustrated in figure 5.1. CrossPhire incorporates three sub-neural networks, each specializing in extracting deep features from each data source. CrossPhire captures both the textual and visual essence of the website by fusing feature logits generated by (i) the GramBeddings model [2] using URLs, (ii) fine-tuned pre-trained image models using the screenshot image, and (iii) pre-trained Sentence Transformer models using the extracted textual content.

In the context of phishing website identification, the websites are classified into two categories: phishing and legitimate. We aim to distinguish between phishing and legitimate websites, making it a binary classification problem. Hence, we train the network to produce the value '1' if the input is phishing and '0' otherwise. In the following steps, we discuss the architecture of CrossPhire and how it processes each data modality by exploring its individual components. Finally, we discuss combining the subnetworks and the joint training procedure.

5.2.1. URL Processing

As previously discussed, the web page's URL contains essential information commonly used to derive meaningful features to differentiate between phishing and legitimate web pages. For URL feature representation, we consider the GramBeddings model proposed in [2] to generate contextualized embeddings capturing the essential features of the URL. We mainly opted for GramBeddings over other potential URL-based models because Grambeddings outperformed its baseline approaches in their original study [2], demonstrating high robustness and generalization capability while obtaining state-of-the-art results. Additionally, we conduct our own experiments and further analyze the potential anti-phishing URL models using the benchmark datasets in the Experimental Results section.

GramBeddings: The GramBeddings model concatenates character level (unigrams) and n-gram level (i.e., 4, 5, and 6) features to create feature embeddings from URL strings. For

Lower Case	abcdefghijklmnopqrstuvwxyz
Upper Case	ABCDEFGHIJKLMNOPQRSTUVWXYZ
Numerical	0123456789
Special Characters	,;.!”?:’/\\|_@#$%□&*~‘+-=<>()[]{}

Figure 5.2 The characters (Alphabet) considered by GramBeddings for URL encoding (adopted from [2]).

Character-level embeddings, the input URL strings are first tokenized and then encoded by replacing each character with its numerical representation based on the alphabetical order, where the first alphabetical character a = 1 and ”/” = 94 (figure 5.2 showcases the considered characters). The encoded vectors less than 128 are padded with zeros and truncated otherwise. For N-gram feature selection, the authors used the chi-square method to overcome the curse of dimensionality problem as the number of possible n-gram combinations rapidly grows. By experimenting with different n values (3-7), the authors obtained the best performance using the values (4, 5, 6). Thus, a feature mapping vector is created for every n value (1, 4, 5, 6), and identical sub-networks are implemented to generate independent feature vectors. Every sub-network consists of two cascaded convolutional layers followed by a Spatial Dropout to minimize adjacent n-gram correlations. The output of the CNN layers is fed into two bidirectional-LSTM (BiLSTM) layers to capture sequential relationships in both directions. Next, an additive attention layer was implemented to benefit from selective parts of the BiLSTM layer output vectors. Finally, the output from every sub-network (i.e., 256×4 sub-networks = 1024) is concatenated to form a 1024 feature logit incorporating the features from the different n-gram selections. Two cascaded hidden layers were employed in the final fully connected layer, reducing the resultant feature vector to 256. In this thesis, we utilize the GramBeddings architecture to generate contextualized embeddings from Raw URLs. Figure 5.3 overviews the overall architecture of GramBeddings, and for a more detailed explanation, you can refer to their original publication [2].

5.2.2. Screenshot Processing: Towards Visual Essence

Since phishers try to deceive users with websites that are visually similar to legitimate websites, suspicious users usually inspect the web page’s appearance to determine its

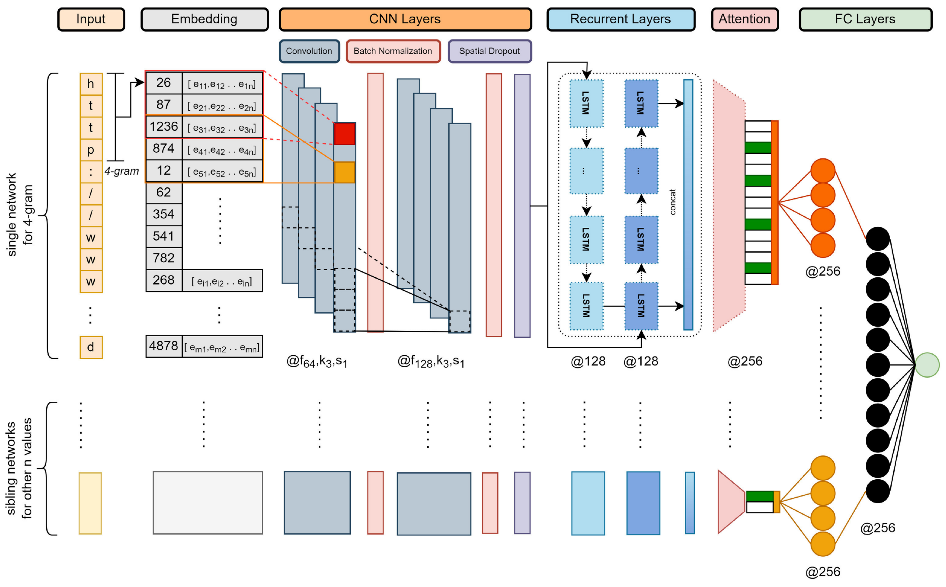


Figure 5.3 Overview of the GramBeddings neural network (adopted from [2]).

authenticity. The visual appearance of a site retains crucial factors like the design quality, formatting, and logos that distinguish phishing from legitimate websites. However, capturing the image's low, mid, and high-level features is challenging.

Thanks to the recent breakthroughs led by deep convolutional neural networks in multiple visual recognition and image classification tasks. In particular, very deep image models (with +100 layers) like ResNets and DenseNet achieved state-of-the-art performance on challenging datasets like ImageNet [81] and COCO [82]. This achievement was possible by increasing the number of layers while maintaining the computational efficiency with a smart approach to mitigate the well-known issue of exploding/vanishing gradients, thereby effectively surpassing the 100-layer barrier [3, 95].

The works in the computer vision domain demonstrated the effectiveness and importance of transfer learning from large pre-train models, particularly those trained models on the ImageNet dataset, which are then fine-tuned for the specific task needed [96]. In our approach, we leverage these large image models and fine-tune them for the phishing identification task, enabling us to capture comprehensive visual features. Unlike existing vision-based methods that rely on partial regions of the image, CrossPhire utilizes the entire screenshot, providing a more holistic analysis.

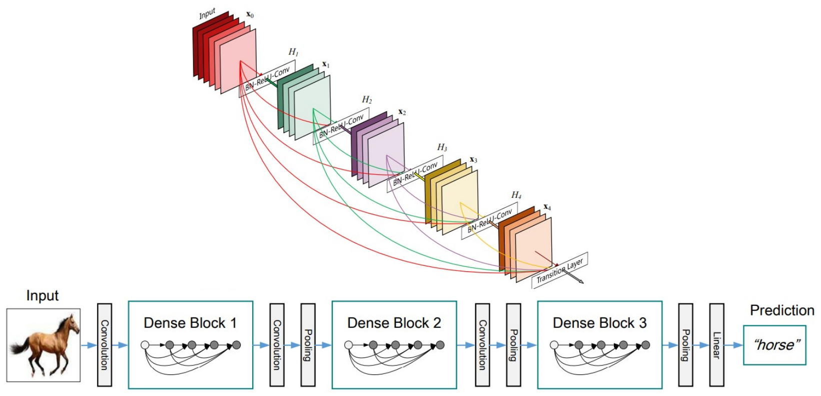


Figure 5.4 Overview of the DenseNet model architecture [3].

Residual Networks [95]: ResNets are sophisticated deep convolutional neural network architectures famous for their depth and strong learning abilities. The network consists of a series of residual blocks that contain multiple convolutional layers with skip connections. The skip connections are one of the key features of ResNets that enable efficient training by providing a shorter path for gradients to propagate, mitigating the vanishing gradient problem. The ResNet architecture also addresses the degradation problem, where adding more layers to the neural network degrades performance by implementing skip connections (shortcut connections). The shortcut connection skips a couple of residual blocks, and their output is added to the output of the stacked layers (skipped layers).

Densely Connected Convolutional Network [3]: DenseNet is a deep convolutional neural network architecture known for its dense connectivity pattern, which facilitates feature reuse and encourages future propagation. Unlike traditional CNNs, DenseNet connects each layer to every other layer in a feed-forward fashion within a dense block, as demonstrated in figure 5.4. As a result, this connectivity pattern solves the vanishing gradient problem and creates shorter paths for better gradient flow during training. Unlike ResNets, where identity mapping adds output feature maps, DenseNet combines the feature maps by concatenating them.

Notably, ResNet and DenseNet excel in extracting deep and sophisticated features from input images and capturing complex visual patterns and structures, which makes them an ideal choice for tasks that require sophisticated image analysis, such as image classification.

Given the high performance of both DenseNet-121 and ResNets-50, We leverage their ability to learn rich representation by inputting the website's screenshot image and capturing visual and structural features to distinguish phishing from legitimate websites. We apply transfer learning rather than training the models for phishing classification from scratch. This process involves utilizing the pre-trained ResNet50 and DenseNet121 models on the ImageNet dataset and fine-tuning them for the phishing classification task using the training images from the benchmark datasets.

5.2.3. Markup-Free Textual Content Processing: Towards Textual Essence

The website's main content includes rich textual content that can be utilized to extract informative textual features. While some previous studies have utilized the source code as a primary feature source to distinguish between phishing and legitimate web pages, none have considered using plain, markup-free, and concise text from the website's source code without further processing steps. These processing steps might result in textual information loss, failing to preserve and capture the semantic relationships within the text. Consequently, we adopted a convenient method to extract the website's *main* content, ensuring that the text from different HTML tags, such as the title, paragraph, and heading tags, is included. We employed several HTML parsers like BeautifulSoup [89] and Trafilatura [88] to extract all the available text in the web page's source code.

For textual feature representation, we employed the well-known state-of-the-art sentence transformer models to convert the extracted text into a numerical representation that captures the context and semantic meaning of the sentences. Despite the widespread use of Large Language Models (LLMs) trained on massive text corpora in various NLP applications, they have not been previously applied in phishing website detection.

Unlike context-independent word embeddings that are generated by processing single words, sentence transformer models leverage the transformer architecture and self-attention mechanism to create sentence embeddings that capture the semantic meaning, order, and context of words within an entire sentence [4]. The transformers use self-attention

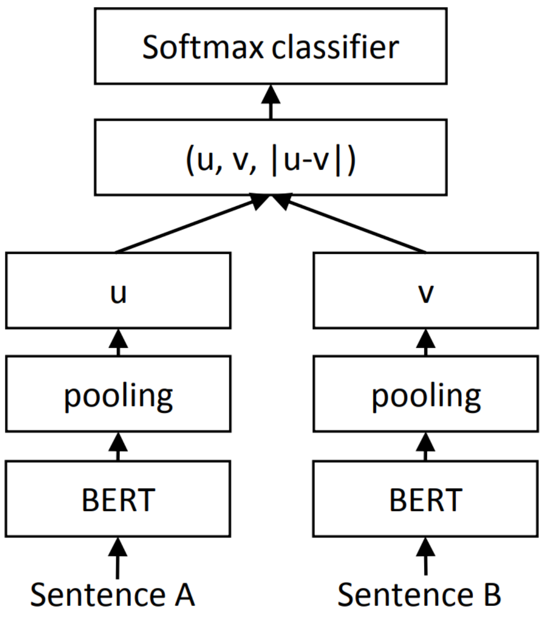


Figure 5.5 Overview of the Sentence-BERT model architecture [4].

mechanisms that allow the model to assign weights to each word in the sentence. The model captures the long-range dependencies and relationships within a sentence by computing a weight representing the importance of different parts of the input sequence.

The Sentence-BERT model, introduced in [4], modifies the BERT model [96] by employing a siamese and triplet network architecture. This modification aims to generate semantically meaningful sentence embeddings that are useful for various sentence-pair tasks like question answering and semantic textual similarity (STS) [4]. Unlike BERT's cross-encoder setup, where two sentences are fed to the transformer for target value prediction, Sentence-BERT incorporates two BERT models with tied weights (siamese network) along with a pooling layer and a softmax classifier, as depicted in Figure 5.5.

Given the plenitude of websites in foreign languages on the internet, the website's extracted plain text comes in numerous languages. Therefore, our approach needs to handle non-English text as well. To achieve that, we employ two different methods to process the extracted textual content. The first method utilizes a multilingual ST model trained on 100 different languages [97]. This model generates similar embedding vectors for semantically similar texts across different languages. The second method involves translating all non-English text into English and then using a monolingual ST model to process the English-translated text. The translation process allows us to compare the performance between the monolingual and multilingual ST models. Consequently, we picked the monolingual MPNet model [98] and the multilingual XLM-RoBERTa model [97], both publicly available on the HuggingFace platform.

In contrast to how we integrated the pre-trained image models by fine-tuning them to the phishing classification task. We utilize the pre-trained sentence transformer models as encoders without fine-tuning. Both MPNet and XLM-R models are employed to produce a fixed-size sentence vector of 768 for each sentence having a maximum of 512 tokens.

5.3. Combining Subnetworks

After generating deep features from the URL, image, and textual content, the different encoders output the logits (embedding vectors) for each data modality. Following the embedding generation, We implemented several hidden layers composed of dense layers to reduce the dimension of the vectors. Next, we fuse the reduced feature vectors (URL-256, image-512, and text-16) by concatenating them into a single vector with a dimension of 784. Lastly, we add fully connected (FC) and classification layers.

5.4. Evaluation Metrics

In this subsection, we provide a brief description and a definition of the various evaluation metrics used in this thesis. These metrics are calculated and used to reflect and evaluate our approach's performance. Additionally, understanding these evaluation methods leads to correct performance interpretation and investigating the advantages and drawbacks of our approach. We first start with introducing the Confusion Matrix as the used evaluation metrics depend on it.

The Confusion Matrix in binary classification problems is a two-dimensional matrix that summarizes the classifier's performance. As demonstrated in table 5.1, the binary confusion matrix consists of two classes, one designated as the positive class and the negative class [99]. In our case, the positive class is the Phishing class, and the negative is the Legitimate. The four cells in the confusion matrix in table 5.1 represent True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN).

	Predicted Positive	**Predicted Negative**
Actual Positive	TP	FN
Actual Negative	FP	TN

Table 5.1 Binary Class Confusion Matrix

The definitions of these values are [99]:

- **True Positive (TP)**: The number of samples that are positive (Phish), and the model correctly classified it as positive.
- **False Positive (FP)**: The number of samples classified as positive (Phish) when they are actually negative (Legit).
- **True Negative (TN)**: The number of samples that are negative (Legit), and the model correctly classified it as negative.
- **False Negative (FN)**: The number of samples classified as negative (Legit) when they are actually positive (Phish).

It is necessary to consider multiple evaluation metrics to obtain a comprehensive understanding of a specific model. In consequence, we calculate and report the values of accuracy, precision, recall, F1-score, False Positive Rate (FPR), False Negative Rate (FNR), and True Negative Rate (TNR). This wide range of metrics provides a detailed summary of the proposed model's performance [100].

The definitions and the way to calculate these metrics are:

- **Accuracy**: The proportion of correctly classified instances out of the total number of instances.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

  The accuracy equation can be explained as follows:

$$\text{Accuracy} = \frac{\text{Total number of correctly classified website samples}}{\text{Total number of website samples}} \tag{2}$$

- **Precision** (positive predictive value): The proportion of correctly classified positive instances out of the total predicted positive instances. This value reflects how accurately the positive class is predicted [101].

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3}$$

  The precision equation can be explained as follows:

$$\text{Precision} = \frac{\text{Number of correctly classified phishing samples}}{\text{Total number of samples predicted as phishing}} \tag{4}$$

- **Recall** (Sensitivity): The proportion of correctly classified positive instances out of the total actual positive instances. This value reflects how well the model avoids missing positive samples [101].

$$\text{Recall} = \frac{TP}{TP + FN} \tag{5}$$

  The recall equation can be explained as follows:

$$\text{Recall} = \frac{\text{Number of correctly classified phishing samples}}{\text{Total number of actual phishing samples}} \tag{6}$$

- **F1-score**: is a combination of Recall and Precision that quantifies the accuracy of identifying positive samples (precision) while also avoiding the misclassification of negative samples (FP).

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{7}$$

- **False Positive Rate (FPR)**: The proportion of negative samples (legit) that are misclassified as positive (phish).

$$\text{FPR} = \frac{FP}{TN + FP} \tag{8}$$

- **False Negative Rate (FNR)**: The proportion of positive samples (phish) that are misclassified as negative (legit).

$$\text{FNR} = \frac{FN}{TP + FN} \tag{9}$$

- **True Negative Rate (TNR)**: (Specificity) The proportion of correctly classified negative instances out of the total predicted negative instances. This value reflects how accurately the negative class is predicted.

$$\text{TNR} = \frac{TN}{TN + FP} \tag{10}$$

  The Specificity equation can be explained as follows:

$$\text{Specificity (TNR)} = \frac{\text{Number of correctly classified legitimate samples}}{\text{Total number of actual legitimate samples}} \tag{11}$$

6. EXPERIMENTAL RESULTS

In this chapter, we discuss the implementation and results of our proposed multimodal approach, CrossPhire, using five different anti-phishing datasets. In addition to our collected dataset Phish360, we utilize four different benchmarking datasets, bringing the total number of website samples in all datasets combined to approximately 358,000. In contrast to other research that combines multiple anti-phishing datasets, we conduct evaluation experiments on each dataset separately to assess CrossPhire's performance across various datasets. Given that the datasets employed in this thesis were collected from diverse sources spanned in different years (from 2016 to 2020), it is crucial to evaluate CrossPhire's effectiveness on each dataset individually to investigate its adaptability to a wide range of phishing tactics. Furthermore, we analyze CrossPhire's generalization and robustness capabilities by conducting numerous out-of-sample evaluation experiments. In these experiments, we train CrossPhire on one dataset's training proportion and test it on another dataset's testing proportion.

6.1. Experimental Setup

In this work, we implemented our proposed approach using Python 3 on the Keras platform. To ensure computational reproducibility and facilitate the reuse and adoption of CrossPhire, we publicly share our codebase and datasets with the research community. During training, we configured CrossPhire with a batch size of 32. We employed the Adam optimizer with an initial learning rate of 0.001. We implemented a cosine annealing learning rate schedule to adjust the learning rate during training dynamically. Finally, we utilized the binary cross-entropy function to monitor the model's loss.

The number of training epochs varied based on the data source for the experiment. When the training and testing data originated from the same source, we trained CrossPhire for 20 epochs. For cross-dataset evaluations, where training and testing data came from different sources, we extended the training duration to 30 epochs. Another critical factor, often overlooked, is the train-test ratio. Although there isn't a clear rule on splitting the datasets, most anti-phishing research splits the data using an 80:20 or 70:30 train test ratio [102]. We divided the data into **80%** training and **20%** testing proportions for all evaluation experiments, setting *random_state* = **42** to ensure consistent sets every time the data is split. We evaluated CrossPhire's performance using the most commonly used metrics of Accuracy, Precision, Recall, and F1-score (described in section Evaluation Metrics).

6.2. Single Modal Based Assessment

In this stage, we conduct several experiments to assess the individual components of CrossPhire. To validate the single-modal components, we employ the benchmarking datasets to experiment with each model and assess its generalization capability. Based on our literature review, we identified suitable models that are compatible with our proposed approach achieving state-of-the-art results in the phishing detection task. For URLs, in addition to GramBeddings [2], we consider the URLNet [103], and URL Tran [104] models. Furthermore, we propose utilizing pre-trained ResNet and DenseNet models for image classification by fine-tuning them to the phishing identification task. Lastly, for the extracted textual content, we employ pre-trained Sentence Transformers, specifically MPNet [98] to process the English-translated BS text and XLM-RoBERTa [97] for the original multilingual TF text.

6.2.1. Pure URL Based Assessment

We conduct comprehensive experiments using the URL samples from benchmark datasets to assess the capabilities of selected models in phishing URL classification. As previously mentioned, attackers constantly adjust their methods and develop new tactics to evade detection systems. Consequently, the collected URL phishing samples from different years within the benchmark datasets naturally encompass various phishing tactics. Therefore, in addition to the traditional same dataset evaluation, a robust method to evaluate the

Table 6.1 In-dataset and cross-dataset benchmarking using different portions of several datasets. The table presents the training and testing accuracies of three models (*GramBeddings*, *URLNet*, and *URL Tran*) to examine the models' same-dataset and cross-dataset performance. Each cell contains the model's accuracy using the corresponding train and test sets. Bold entries highlight the best performance achieved within each experiment.

Data Modality	**Training Dataset**	**Testing Dataset**	**GramBeddings**		**URLNet**		**URL Tran**	
			Train Acc.	**Test Acc.**	Train Acc.	**Test Acc.**	Train Acc.	**Test Acc.**
URL	PILWD-134K	PILWD-134K	0.9832	**0.9691**	0.9878	0.9635	0.9562	0.8984
URL	PILWD-134K	VanNL126k	0.9834	**0.9155**	0.9878	0.9146	0.9551	0.8398
URL	PILWD-134K	PhishIntention	0.9840	0.9308	0.9756	**0.9533**	0.9547	0.7306
URL	PILWD-134K	PWD2016	0.9842	0.9107	0.9922	**0.9528**	0.9546	0.5420
URL	VanNL126k	PILWD-134K	0.9994	0.5741	0.9773	**0.5808**	0.9825	0.5283
URL	VanNL126k	VanNL126k	0.9994	0.9412	0.9773	**0.9852**	0.9716	0.9840
URL	VanNL126k	PhishIntention	0.9999	0.6015	0.9886	0.5510	0.9844	**0.6227**
URL	VanNL126k	PWD2016	0.9993	**0.6738**	0.9773	0.5420	0.9851	0.4915
URL	PhishIntention	PILWD-134K	0.9990	**0.6807**	0.9765	0.6467	0.9883	0.6624
URL	PhishIntention	VanNL126k	0.9981	**0.9015**	0.9922	0.8987	0.9871	0.7587
URL	PhishIntention	PhishIntention	0.9994	**0.9900**	1.0000	0.9889	0.9879	0.9363
URL	PhishIntention	PWD2016	0.9996	0.9937	0.9766	**0.9985**	0.9867	0.8948
URL	PWD2016	PILWD-134K	0.9995	**0.6515**	1.0000	0.5000	0.9942	0.5526
URL	PWD2016	VanNL126k	1.0000	**0.7940**	1.0000	**0.7940**	0.9999	0.7359
URL	PWD2016	PhishIntention	1.0000	0.6717	1.0000	0.5410	1.0000	**0.8581**
URL	PWD2016	PWD2016	0.9933	0.9948	1.0000	**1.0000**	1.0000	0.9983

performance of the selected URL phishing identification models is to conduct cross-dataset evaluations.

In these experiments, we train the models using the training set of one dataset and test them on the testing set of another dataset. This approach allows us to explore the generalization capability of the models in identifying URL phishing samples collected in different years, encompassing a wide range of phishing tactics to evade detection systems. Table 6.1 presents a comparison between the performance of the selected models. To ensure comparability, we used the same train-test split of 80% training and 20% testing with a seed_value of 42. The bold entries in table 6.1 highlight the highest test accuracy across the models employed (row-wise).

Notably, the results in the table contain both same-dataset experiments, where the model is trained and tested using the same dataset, and cross-dataset evaluation, where the model is trained on one dataset's train set (80%) and tested on another dataset's test set (20%). Examining the testing accuracies in table 6.1, it is evident that GramBeddings outperforms the other models (URLNet and URL Tran) by achieving higher testing accuracy in the majority of the experiments (8 out of 16). Based on these findings, we integrated the GramBeddings model to process the URLs in our proposed approach, CrossPhire.

6.2.2. Pure Screenshot Based Assessment

Similar to our experimental method applied using the websites' URL samples, we conducted several experiments utilizing pre-trained ResNet50 and DenseNet121 models, originally trained on the ImageNet dataset. In these experiments, we leveraged the screenshot images available in the multimodal datasets to fine-tune the image models for the phishing classification task.

We employed both traditional same-dataset evaluation and cross-dataset evaluation techniques to assess the robustness of the selected models. Rather than training the image models from scratch, we opted for transfer learning. This involved utilizing the trained weights from the pre-trained models on the large ImageNet dataset, which contains approximately 14.2 million images. Similar to the train-test split method in previous experiments, we fine-tuned these models using the screenshot images in the train set (80%) and tested their performance on the test set (20%). Table 6.2 presents a comparison between the image models using the benchmark datasets. Examining the testing accuracies in table 6.2, it is evident that the DenseNet121 model outperforms the ResNet50 in phishing image classification. Due to the minor difference in accuracies achieved by both models, we opted to include both image models in our CrossPhire evaluations for further experimentation.

6.2.3. Pure Textual Content Assessment

In these experiments, we utilize two types of extracted textual content, namely Trafilatura_text (TF) and BeautifulSoup_text (BS), to classify web pages as phishing or legitimate. Since the extracted text is multilingual, we employ the multilingual XLM-R sentence transformer, namely 'aditeyabaral/sentencetransformer-xlm-roberta-base', available on the HuggingFace community. The model takes a maximum of 512 tokens as

Table 6.2 Single-modal in-dataset and cross-dataset evaluations of image models (ResNet and DenseNet) using web page screenshot image across benchmarking datasets

Data Modality	Image Model	Training Dataset	Testing Dataset	Test Acc.	Test F1
Screenshot	Resnet50	**PILWD-134K**	**PILWD-134K**	0.9304	0.9248
Screenshot	Densenet121	**PILWD-134K**	**PILWD-134K**	**0.9562**	**0.9544**
Screenshot	Resnet50	PILWD-134K	VanNL126K	0.7309	0.8201
Screenshot	Densenet121	PILWD-134K	VanNL126K	0.7848	0.8569
Screenshot	Resnet50	PILWD-134K	PhishIntention	0.6392	0.6963
Screenshot	Densenet121	PILWD-134K	PhishIntention	0.7035	0.7350
Screenshot	Resnet50	PILWD-134K	PWD2016	0.6882	0.6740
Screenshot	Densenet121	PILWD-134K	PWD2016	0.6946	0.6547
Screenshot	Resnet50	PILWD-134K	Phish360	0.7445	0.5995
Screenshot	Densenet121	PILWD-134K	Phish360	0.7887	0.7064
Screenshot	Resnet50	VanNL126K	PILWD-134K	0.6457	0.6939
Screenshot	Densenet121	VanNL126K	PILWD-134K	0.6704	0.7088
Screenshot	Resnet50	**VanNL126K**	**VanNL126K**	0.9392	0.9627
Screenshot	Densenet121	**VanNL126K**	**VanNL126K**	**0.9577**	**0.9733**
Screenshot	Resnet50	VanNL126K	PhishIntention	0.6938	0.7395
Screenshot	Densenet121	VanNL126K	PhishIntention	0.7266	0.7633
Screenshot	Resnet50	VanNL126K	PWD2016	0.6501	0.5852
Screenshot	Densenet121	VanNL126K	PWD2016	0.7088	0.6264
Screenshot	Resnet50	VanNL126K	Phish360	0.8646	0.8303
Screenshot	Densenet121	VanNL126K	Phish360	0.8753	0.8413
Screenshot	Resnet50	PhishIntention	PILWD-134K	0.6438	0.6621
Screenshot	Densenet121	PhishIntention	PILWD-134K	0.6910	0.6745
Screenshot	Resnet50	PhishIntention	VanNL126K	0.6081	0.7005
Screenshot	Densenet121	PhishIntention	VanNL126K	0.6615	0.7497
Screenshot	Resnet50	**PhishIntention**	**PhishIntention**	0.9556	0.9564
Screenshot	Densenet121	**PhishIntention**	**PhishIntention**	**0.9703**	**0.9696**
Screenshot	Resnet50	PhishIntention	PWD2016	0.7121	0.6778
Screenshot	Densenet121	PhishIntention	PWD2016	0.6813	0.5950
Screenshot	Resnet50	PhishIntention	Phish360	0.7301	0.5156
Screenshot	Densenet121	PhishIntention	Phish360	0.7720	0.6099
Screenshot	Resnet50	PWD2016	PILWD-134K	0.5720	0.5474
Screenshot	Densenet121	PWD2016	PILWD-134K	0.5858	0.3959
Screenshot	Resnet50	PWD2016	VanNL126K	0.7127	0.8077
Screenshot	Densenet121	PWD2016	VanNL126K	0.7733	0.8616
Screenshot	Resnet50	PWD2016	PhishIntention	0.6803	0.6151
Screenshot	Densenet121	PWD2016	PhishIntention	0.6796	0.6673
Screenshot	Resnet50	**PWD2016**	**PWD2016**	**0.9321**	**0.9329**
Screenshot	Densenet121	**PWD2016**	**PWD2016**	0.9309	0.9294
Screenshot	Resnet50	PWD2016	Phish360	0.6952	0.5036
Screenshot	Densenet121	PWD2016	Phish360	0.7245	0.6530
Screenshot	Resnet50	Phish360	PILWD-134K	0.6773	0.6760
Screenshot	Densenet121	Phish360	PILWD-134K	0.6985	0.7132
Screenshot	Resnet50	Phish360	VanNL126K	0.8589	0.9116
Screenshot	Densenet121	Phish360	VanNL126K	0.8668	0.9153
Screenshot	Resnet50	Phish360	PhishIntention	0.7492	0.7753
Screenshot	Densenet121	Phish360	PhishIntention	0.7601	0.7801
Screenshot	Resnet50	Phish360	PWD2016	0.7014	0.6952
Screenshot	Densenet121	Phish360	PWD2016	0.7255	0.7308
Screenshot	Resnet50	**Phish360**	**Phish360**	0.9386	0.9200
Screenshot	Densenet121	**Phish360**	**Phish360**	**0.9432**	**0.9263**

Table 6.3 Comparison of phishing identification performance using the text extracted with Trafilatura (TF) and BeautifulSoup (BS) for Phish360 dataset. Train and test accuracy scores are reported for Support Vector Machine (SVM), XGBoost (XGB), and CatBoost (CatB) classifiers.

Dataset	**Text**	**SVM**		**XGB**		**CatB**	
		Train Acc.	**Test Acc.**	Train Acc.	**Test Acc.**	Train Acc.	**Test Acc.**
Phish360	Original BS text	0.8663	**0.8658**	0.9995	**0.9798**	0.9970	**0.9772**
Phish360	Original TF text	0.8005	0.7980	1.0000	0.9403	0.9970	0.9356

input and outputs a 768-dimensional vector for each input. So, to compare the generated embeddings using the two different extracted texts, we use the phish360 dataset and implement conventional ML models to classify the textual content into two classes: phishing and legitimate.

The generated embedding vectors are used as feature inputs to some widely utilized ML models in text classification. Specifically, we employed the Support Vector Machine (SVM), XGBoost classifier (XGB), and CatBoost classifier (CatB). We implemented the ML algorithms with their default parameters and a linear kernel we selected for SVM. Using the same train-test split and seed value with the previous experiments, we showcase the results in table 6.3. Examining the results in table 6.3, it is apparent that BS text is more comprehensive compared to the TF text, resulting in a superior feature representation and higher performance across the same models.

Given the superior performance achieved with the BS text, we translate the BS text for all the benchmark datasets to evaluate our method and explore the performance difference between monolingual and multilingual sentence transformers on original and English-translated text. We selected the MPNet monolingual sentence transformer, namely 'sentence-transformers/all-mpnet-base-v2', available on the HuggingFace community. Similar to XLM-R, MPNet also takes 512 tokens as input and produces a 768-dimensional vector as output.

The results for all the datasets are presented in table 6.4, revealing that the MPNet model provides better textual encoding using the English text compared to XLM-R with multilingual text. These findings highlight the significance of the chosen sentence

Table 6.4 Comparison of phishing identification performance across benchmark datasets using the original and translated BeautifulSoup (BS) text with sentence transformer models. Train and test accuracy scores are reported for Support Vector Machine (SVM), XGBoost (XGB), and CatBoost (CatB) classifiers.

Dataset	**Text**	**Transformer**	**SVM**		**XGB**		**CatB**	
			Train Acc.	**Test Acc.**	Train Acc.	**Test Acc.**	Train Acc.	**Test Acc.**
PhishIntention	Translated BS	MPNet	0.9707	**0.9687**	0.9994	**0.9860**	0.9984	**0.9838**
PhishIntention	Original BS	XLM-R	0.8663	0.8658	0.9995	0.9798	0.9970	0.9772
VanNL126k	Translated BS	MPNet	0.9611	**0.9600**	0.9982	0.9749	0.9932	**0.9744**
VanNL126k	Original BS	XLM-R	0.9027	0.9023	0.9975	**0.9754**	0.9919	0.9734
PILWD-134K	Translated BS	MPNet	0.9211	**0.9184**	0.9913	**0.9550**	0.9801	**0.9563**
PILWD-134K	Original BS	XLM-R	0.7946	0.7969	0.9882	0.9529	0.9772	0.9512
PWD2016	Translated BS	MPNet	0.9599	0.9528	0.9996	**0.9899**	0.9993	**0.9882**
PWD2016	Original BS	XLM-R	0.9739	**0.9614**	0.9995	0.9882	0.9993	0.9862
Phish360	Translated BS	MPNet	0.9641	**0.9622**	1.0000	**0.9716**	1.0000	**0.9702**
Phish360	Original BS	XLM-R	0.8421	0.8531	1.0000	0.9571	0.9979	0.9590

transformer and its impact on the overall performance. This outcome was expected, given that the monolingual transformer was pre-trained on a larger English corpus than the multilingual transformer.

6.3. CrossPhire: Multimodal Detection Model Evaluations

After the extensive single-modal experiments validating our design choice in URL and image models, we detail the experiments using the textual extracted content. As it is known that monolingual LLM outperform multilingual ones, we conduct two main experiments with two distinct text extraction methods using Trafilatura (TF) and BeautifulSoup (BS). As a baseline, we used the original extracted TF text containing numerous languages with the XLM-R multilingual Sentence Transformer (ST), as illustrated in figure 6.1; we refer to this experiment as **CP-EX-1**. Since we have established that BS text is more comprehensive and more detailed than TF text, we opted for translating the BS text into English to explore the potential difference in the generated textual feature embeddings. Hence, we use the translated BS text with the monolingual ST, referring to this experiment as **CP-EX-2** (see figure 6.1).

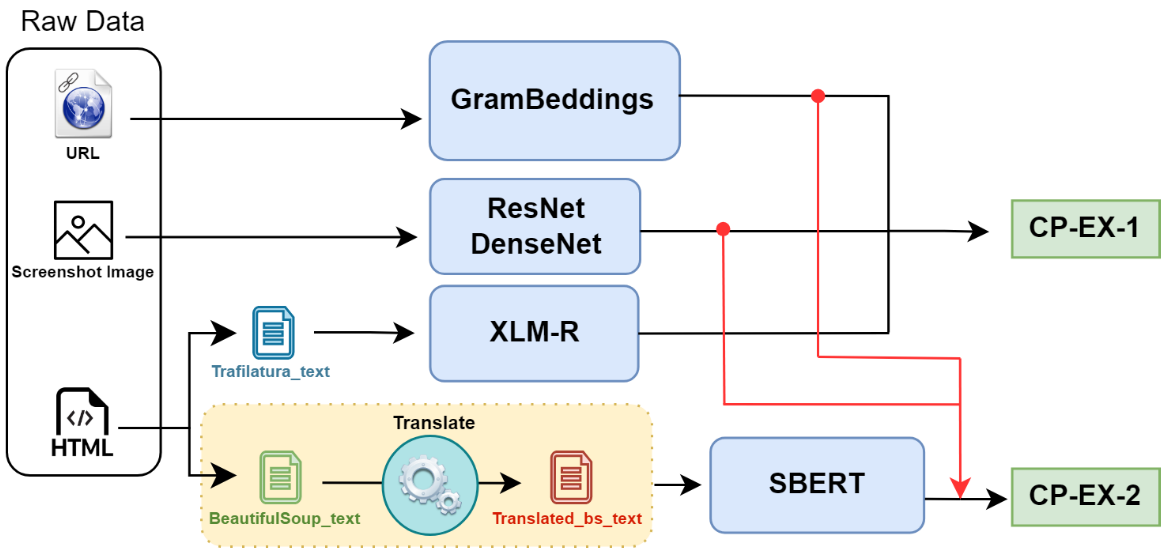


Figure 6.1 Comparison between the two main configuration (1) CrossPhire-Experiment-1 with original trafilatura text and (2) CrossPhire-Experiment-2 with English-translated BeautifulSoup text

6.3.1. CP-EX-1: CrossPhire With Multilingual Text

This experiment employs the GramBeddings model for URLs, ResNet50 and DenseNet121 for screenshot images, and the multilingual XLM-R sentence transformer ('aditeyabaral/sentencetransformer-xlm-roberta-base' available on the HuggingFace community) model with the original Trafilatura text. The results are presented in table 6.8. CrossPhire (EX-1) achieves high testing accuracy and F1 scores across all datasets with both image model components. Notably, across the benchmark datasets, both ResNet and DenseNet achieve comparable results, indicating no significant difference in their performance. However, Phish360 presents a challenge, as the performance of both CrossPhire variants declines compared to the benchmark datasets.

6.3.2. CP-EX-2: CrossPhire With English Text

This experiment employs the GramBeddings model for URLs, ResNet50 and DenseNet121 for screenshot images, and the monolingual BERT-like sentence transformer ('sentence-transformers/all-mpnet-base-v2' available on the HuggingFace community)

Table 6.5 CP-EX-1: Assessing CrossPhire's performance across multimodal datasets using multilingual Trafilatura text

Image Model	**Dataset**	**Train Acc.**	**Test Acc.**	**Test Precision**	**Test Recall**	**Test F1**
Resnet50	PILWD-134K	0.9904	**0.9812**	0.9849	0.9763	0.9791
Densenet121	PILWD-134K	0.9804	0.9798	0.9805	0.9779	0.9782
Resnet50	VanNL126K	0.9933	0.9904	0.9953	0.993	0.9941
Densenet121	VanNL126K	0.9919	**0.9909**	0.9940	0.9948	0.9948
Resnet50	PhishIntention	0.9973	0.9945	0.9935	0.9972	0.9949
Densenet121	PhishIntention	0.9963	**0.9955**	0.9962	0.9960	0.9959
Resnet50	PWD2016	1.000	**1.000**	1.000	1.000	1.000
Densenet121	PWD2016	1.000	**1.000**	1.000	1.000	1.000
Resnet50	Phish360	0.9977	**0.9748**	0.9765	0.9606	0.9556
Densenet121	Phish360	0.9963	0.9687	0.9523	0.9711	0.9495

Table 6.6 CP-EX-2: Assessing CrossPhire's performance across multimodal datasets using the English-translated BeautifulSoup text

Image Model	**Dataset**	**Train Acc.**	**Test Acc.**	**Test Precision**	**Test Recall**	**Test F1**
Resnet50	PILWD-134K	0.9980	0.9804	0.9788	0.9810	0.9783
Densenet121	PILWD-134K	0.9974	**0.9807**	0.9784	0.9821	0.9883
Resnet50	VanNL126K	0.9995	**0.9942**	0.9957	0.9972	0.9963
Densenet121	VanNL126K	0.9919	0.9926	0.9949	0.9962	0.9952
Resnet50	PhishIntention	0.9981	0.9957	0.9950	0.9976	0.9961
Densenet121	PhishIntention	0.9990	**0.9963**	0.9954	0.9983	0.9966
Resnet50	PWD2016	1.000	**1.000**	1.000	1.000	1.000
Densenet121	PWD2016	1.000	**1.000**	1.000	1.000	1.000
Resnet50	Phish360	0.9991	**0.9921**	0.9818	0.9988	0.9835
Densenet121	Phish360	0.9988	0.9796	0.9790	0.9699	0.9653

model with the translated BeautifulSoup text. The results are presented in table 6.6. CrossPhire (EX-2) with the English-translated BS text outperforms the models trained on multilingual text (CP-EX-1) with an exception for the PILWD-134k dataset where CP-EX-1 achieved slightly better accuracy and precision scores. Notably, The accuracy on the Phish360 dataset significantly improved from 97.49% with multilingual text to 99.21% with translated English text. This demonstrates the superiority of monolingual sentence transformers trained on English content, which is due to several previously mentioned factors, such as access to larger and higher-quality English datasets.

6.4. Evaluation of CrossPhire Performance and Comparative Analysis

To gain deeper insights into CrossPhire's effectiveness, we explore the answers to critical evaluation questions by analyzing the achieved results between CrossPhire (multimodal) and single-modal alternatives. We further assess CrossPhire's generalization capabilities by analyzing the results of out-of-sample experiments. Then, we finally compare the performance of CrossPhire with its baseline approaches utilizing the same anti-phishing datasets.

6.4.1. Comparing CrossPhire Performance With Single-modal Components

Q1: Does incorporating multiple data sources enhance phishing detection accuracy?

This question investigates the performance of CrossPhire compared to its single-modal components utilizing the following data modalities:

- **URL:** GramBeddings model
- **Image:** ResNet50 model
- **Text:** XGBoost model trained on English BS text embeddings

Figure 6.2 illustrates the accuracy of CrossPhire and the single-modal alternatives. To ensure a fair comparison, CrossPhire-EX-2 (purple bar) utilizes the same architecture as the single-modal models: GramBeddings for URLS, ResNet50 for images, and MPNet for the translated BeatifulSoup text.

As evident from the figure, combining multiple data modalities significantly improves the detection accuracy compared to the best-performing single-modal models across all datasets. This finding aligned with various studies [27, 57, 75, 105], highlighting that combining multiple data modalities (hybridization) increases the performance, creating more robust anti-phishing models. Notably, the image model performs worse than the other single-modal models (GramBeddings and XGBoost).

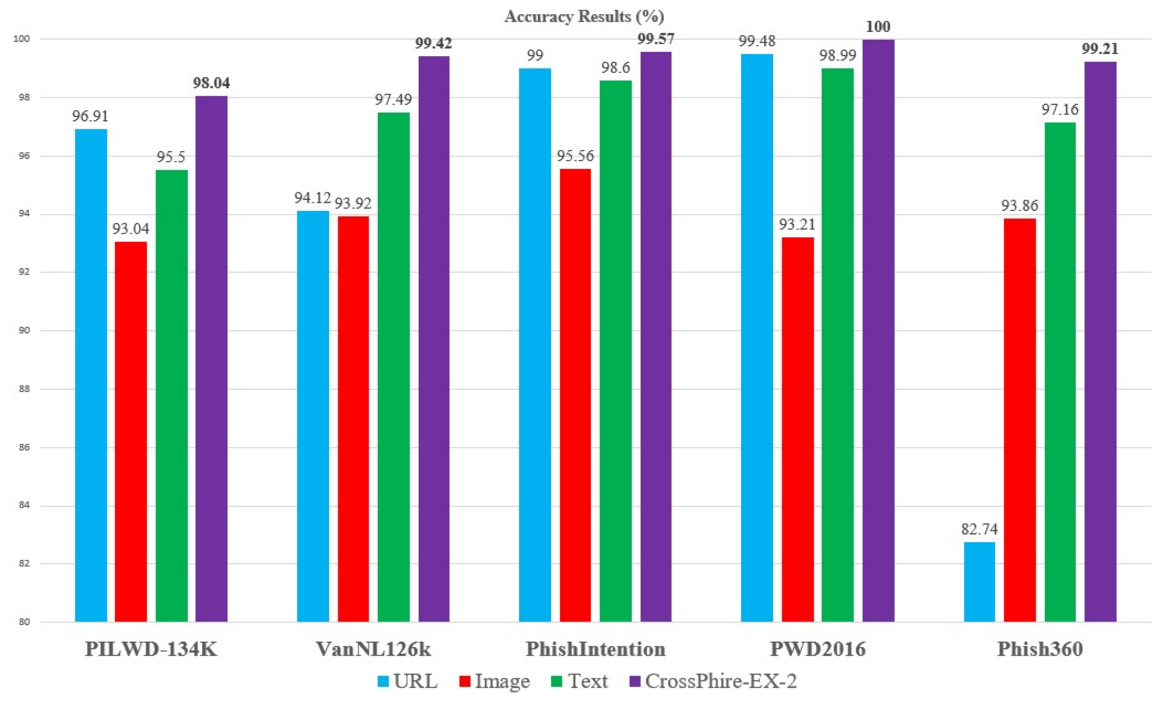


Figure 6.2 Accuracy comparison between CrossPhire and its individual single-modal components on benchmark datasets

6.4.2. CrossPhire Performance in Out-of-Sample Experiments

Q2: How does CrossPhire perform in out-of-sample (cross-dataset) experiments?

This is crucial since the performance of machine learning-based anti-phishing solutions, especially those dependent on hand-crafted features, declines with time since attackers develop new tactics to evade detection systems. When tested on newer phishing samples, the anti-phishing solutions trained on outdated datasets often fail to maintain their original performance. Sánchez-Paniagua et al. [26] also observed the performance decline, where the authors conducted out-of-sample evaluations using URL datasets collected in different years. The authors noted that anti-phishing models trained on outdated datasets struggled to sustain high detection accuracy when tested on recent phishing URLs due to attackers' constant development of phishing tactics.

The authors trained a LightGBM classifier on the PWD2016 and Ebbu2017 datasets, reporting 97.60% and 95.94% testing accuracies. However, when the trained models

Table 6.7 This table presents information on the datasets utilized in the study, including the dataset names, sample collection year, the sample size in each class, and the legitimate and phishing data sources

Dataset Name	Samples Collection Year	Sample Size	Legitimate Data Source	Phishing Data Source
PWD2016	Mar-Apr 2016	Phish: 15,000 Legit: 15,000	Alexa DMOZ BOTW	PhishTank
VanNL126k	Sept-Dec 2019	Phish: 100,000 Legit: 25,938	DMOZ	PhishTank OpenPhish PhishStats
PhishIntention	Oct 2019-Aug 2020	Phish: 29,496 Legit: 25,400 Misleading Legit: 3,049 (Apr. 2021)	Alexa	OpenPhish
PILWD-134k	Aug 2019-Sep 2020	Phish: 66,964 Legit: 66,964	Majestic Million Quantcast	PhishTank
Phish360 (Ours)	2019-2021	Phish: 4,332 Legit: 6,416	Custom crawl Random Subsampling	PhishTank

were tested on a more recently collected dataset in 2020 (PIU-60k), the testing accuracies dropped to 87.18% for the model trained on PWD2016 and 65.25% for the model trained on Ebbu2017. Similarly, Opara et al. [66] argued that the earliest collected data should always be used for training, and the testing should be done on the most recently collected data. For this reason, in their data collection process, the authors collected the training data in November 2018 and testing data in January 2019 to assess their models' robustness and generalizability.

In our experiments, we take it a step further by utilizing the benchmark multimodal datasets to evaluate the robustness and generalization of CrossPhire. As demonstrated in table 6.7, the multimodal data samples were collected in different years, with the oldest dataset dating back to 2016 and the most recent to 2021.

Similar to the evaluation experiments in the earlier subsection CrossPhire Evaluations, we follow the exact configuration of both variants of CrossPhire. The first variant utilizes multilingual trafilatura text with XLM-R ST, and the second one employs English-translated BeautifulSoup text with MPNet.

In both experiments, we train CrossPhire using one dataset's training set (80%) presented in the Train Dataset column in table 6.8 and test it on another dataset's testing set (20%) presented in the Test Dataset column in table 6.8, ensuring consistent sets using the seed_value of 42. We first present the results of CrossPhire with multilingual trafilatura text

Table 6.8 This table presents the out-of-sample evaluations of CrossPhire (CP-EX-1) to asses its generalization capabilities using multilingual Trafilatura text

Image Model	Training Dataset	Testing Dataset	Train Accuracy	Val Accuracy	Val Precision	Val Recall	Val F1
Resnet50	PILWD-134K	VanNL126K	0.9772	0.9252	0.9484	0.9613	0.9534
Densenet121	PILWD-134K	VanNL126K	0.9954	0.9172	0.947	0.9525	0.9482
Resnet50	PILWD-134K	PhishIntention	0.9893	0.9261	0.9206	0.9564	0.9343
Densenet121	PILWD-134K	PhishIntention	0.9886	0.9174	0.929	0.9302	0.9261
Resnet50	PILWD-134K	PWD2016	0.9957	0.9111	0.9479	0.856	0.892
Densenet121	PILWD-134K	PWD2016	0.9934	0.9289	0.9434	0.9014	0.9175
Resnet50	PILWD-134K	Phish360	0.9894	0.9151	0.9231	0.8611	0.8749
Densenet121	PILWD-134K	Phish360	0.9590	0.8861	0.8621	0.8542	0.8395
Resnet50	VanNL126K	PILWD-134K	0.9959	0.6354	0.5746	0.9711	0.7116
Densenet121	VanNL126K	PILWD-134K	0.9914	0.6998	0.6911	0.6949	0.6785
Resnet50	VanNL126K	PhishIntention	0.9865	0.6167	0.6064	0.9866	0.7426
Densenet121	VanNL126K	PhishIntention	0.9939	0.6175	0.6065	0.9895	0.7437
Resnet50	VanNL126K	PWD2016	0.9827	0.4782	0.4782	0.9483	0.628
Densenet121	VanNL126K	PWD2016	0.9963	0.4663	0.464	0.9471	0.6127
Resnet50	VanNL126K	Phish360	0.9973	0.7354	0.6077	0.9699	0.7319
Densenet121	VanNL126K	Phish360	0.9950	0.8068	0.9657	0.9259	0.7791
Resnet50	PhishIntention	PILWD-134K	0.9995	0.7451	0.6898	0.8674	0.7578
Densenet121	PhishIntention	PILWD-134K	0.9958	0.7491	0.6997	0.8507	0.7563
Resnet50	PhishIntention	VanNL126K	0.9947	0.9124	0.9471	0.9462	0.9449
Densenet121	PhishIntention	VanNL126K	0.9971	0.9122	0.9397	0.9544	0.9452
Resnet50	PhishIntention	PWD2016	0.9866	0.9977	1.000	0.995	0.9973
Densenet121	PhishIntention	PWD2016	0.9898	0.9988	0.9988	0.9988	0.9988
Resnet50	PhishIntention	Phish360	0.9960	0.8427	0.7479	0.9201	0.8109
Densenet121	PhishIntention	Phish360	0.9987	0.7867	0.6611	0.9664	0.7675
Resnet50	PWD2016	PILWD-134K	1.000	0.4876	0.4876	1.000	0.6457
Densenet121	PWD2016	PILWD-134K	1.000	0.4876	0.4876	1.000	0.6457
Resnet50	PWD2016	VanNL126K	1.000	0.8215	0.8215	1.000	0.8989
Densenet121	PWD2016	VanNL126K	1.000	0.8215	0.8215	1.000	0.8989
Resnet50	PWD2016	PhishIntention	1.000	0.5899	0.5885	1.000	0.7326
Densenet121	PWD2016	PhishIntention	1.000	0.5873	0.5869	1.000	0.7314
Resnet50	PWD2016	Phish360	1.000	0.4032	0.4032	1.000	0.5617
Densenet121	PWD2016	Phish360	1.000	0.4032	0.4032	1.000	0.5617
Resnet50	Phish360	PILWD-134K	0.9659	0.7925	0.7467	0.8694	0.7935
Densenet121	Phish360	PILWD-134K	0.9789	0.7787	0.7686	0.7814	0.7609
Resnet50	Phish360	VanNL126K	0.9839	0.9181	0.9367	0.9656	0.9492
Densenet121	Phish360	VanNL126K	0.9557	0.9400	0.9560	0.9717	0.9626
Resnet50	Phish360	PhishIntention	0.9729	0.9466	0.9303	0.9824	0.9533
Densenet121	Phish360	PhishIntention	0.9522	0.9054	0.8729	0.9816	0.9201
Resnet50	Phish360	PWD2016	0.9613	0.9101	0.9353	0.9668	0.8904
Densenet121	Phish360	PWD2016	0.9951	0.8826	0.8888	0.8548	0.8590

in table 6.8. Our dataset, Phish360, stands as the smallest among the multimodal datasets, comprising roughly 8,600 training samples. Despite its size, models trained on Phish360 have demonstrated remarkable performance compared to those trained on PWD2016 and VanNL126k. Notably, the model trained on Phish360 has outperformed all others, achieving the highest accuracy of 94.66% when tested on PhishIntention.

Furthermore, CrossPhire trained on Phish360 achieved over 90% accuracy on all datasets except PILWD-134k, which contains around 26,800 testing samples, three times the size of Phish360's training set. The models trained on PWD2016 and VanNL126k showed the poorest results when tested on other datasets, with accuracy scores of as low as 40%. The results in table 6.8 also show that the PILWD-134k dataset challenges the models trained on the other datasets.

Moving on to the evaluation experiments with the second variant of CrossPhire, we maintained the same dataset setup with the same seed value to produce the exact train and test sets. This time, we utilized the English-translated BeautifulSoup text with the MPNet sentence transformer for out-of-sample evaluations. Similar to the earlier experiments with the first variant of CrossPhire, the monolingual setup once again outperformed the multilingual approach, boasting an accuracy gain of approximately 5%, as depicted in table 6.9.

Notably, the model trained on PhishIntention outperformed the others by obtaining an outstanding 99.69% accuracy on the PWD2016 dataset. However, the model trained on Phish360 outperforms PhishIntention's when tested on PILWD-134k, achieving an accuracy of 85.74%, the highest accuracy among all models trained on the other datasets. In the case of the largest benchmark dataset, PILWD-134k, the models trained on it obtained the highest testing accuracies of 93.17% on VanNL126k, 96.27% on PhishIntention, 93.7% on PWD2016, and 93.38% on Phish360.

Overall, our findings highlight CrossPhire's robustness and effectiveness across various datasets, underscoring the importance of dataset selection and model configuration in achieving optimal performance. This is especially noticeable in the case of PWD2016, where the model achieves a perfect 100% across all metrics when trained and tested on PWD2016. However, in the cross-dataset evaluations, the models trained on PWD2016 demonstrated the poorest generalization performance across all the datasets.

6.4.3. Comparing CrossPhire With Baseline Approaches

Q3: Does CrossPhire outperform comparative approaches on benchmark datasets?

Table 6.9 This table presents the cross-dataset evaluations of CrossPhire (**CP-EX-2**) to asses its generalization capabilities using the English-translated BeautifulSoup text

Image Model	**Train Dataset**	**Test Dataset**	**Train Acc.**	**Test Acc.**	**Test Precision**	**Test Recall**	**Test F1**
Resnet50	PILWD-134K	VanNL126K	0.9819	0.9317	0.9644	0.9520	0.9565
Densenet121	PILWD-134K	VanNL126K	0.9723	0.9302	0.9646	0.9499	0.9554
Resnet50	PILWD-134K	PhishIntention	0.9906	0.9511	0.9356	0.9843	0.9573
Densenet121	PILWD-134K	PhishIntention	0.9882	0.9627	0.9568	0.9807	0.9673
Resnet50	PILWD-134K	PWD2016	0.9819	0.9370	0.9471	0.9162	0.9211
Densenet121	PILWD-134K	PWD2016	0.9815	0.9301	0.9221	0.9285	0.9165
Resnet50	PILWD-134K	Phish360	0.9894	0.9338	0.9021	0.9376	0.9068
Densenet121	PILWD-134K	Phish360	0.9713	0.9305	0.8848	0.9514	0.9100
Resnet50	VanNL126K	PILWD-134K	0.9939	0.6840	0.6102	0.9750	0.7406
Densenet121	VanNL126K	PILWD-134K	0.9970	0.7042	0.6269	0.9728	0.7525
Resnet50	VanNL126K	PhishIntention	0.9833	0.6388	0.6193	0.9967	0.7559
Densenet121	VanNL126K	PhishIntention	0.9971	0.6854	0.6781	0.8820	0.7576
Resnet50	VanNL126K	PWD2016	0.9940	0.5168	0.4910	0.9682	0.6403
Densenet121	VanNL126K	PWD2016	0.9920	0.5050	0.4846	0.9542	0.6317
Resnet50	VanNL126K	Phish360	0.9991	0.8494	0.7348	0.9803	0.8243
Densenet121	VanNL126K	Phish360	0.9831	0.8308	0.7061	0.9942	0.8092
Resnet50	PhishIntention	PILWD-134K	0.9972	0.8138	0.7788	0.8635	0.8081
Densenet121	PhishIntention	PILWD-134K	0.9935	0.8061	0.7658	0.8679	0.8022
Resnet50	PhishIntention	VanNL126K	0.9839	0.9069	0.9458	0.9552	0.9278
Densenet121	PhishIntention	VanNL126K	0.9851	0.9155	0.9521	0.9447	0.9465
Resnet50	PhishIntention	PWD2016	0.9973	0.9936	0.9987	0.9876	0.9901
Densenet121	PhishIntention	PWD2016	0.9936	0.9969	0.9988	0.9946	0.9835
Resnet50	PhishIntention	Phish360	0.9994	0.8834	0.7954	0.9572	0.9603
Densenet121	PhishIntention	Phish360	0.9990	0.9380	0.8944	0.9595	0.9119
Resnet50	PWD2016	PILWD-134K	1.000	0.4876	0.4876	1.000	0.6457
Densenet121	PWD2016	PILWD-134K	1.000	0.4876	0.4876	1.000	0.6457
Resnet50	PWD2016	VanNL126K	1.000	0.8214	0.8214	1.000	0.8989
Densenet121	PWD2016	VanNL126K	1.000	0.8214	0.8214	1.000	0.8989
Resnet50	PWD2016	PhishIntention	1.000	0.6084	0.5996	1.000	0.7414
Densenet121	PWD2016	PhishIntention	1.000	0.6125	0.6021	1.000	0.7434
Resnet50	PWD2016	Phish360	1.000	0.4033	0.4033	1.000	0.5602
Densenet121	PWD2016	Phish360	1.000	0.4033	0.4033	1.000	0.5602
Resnet50	Phish360	PILWD-134K	0.9914	0.8582	0.8538	0.8556	0.8456
Densenet121	Phish360	PILWD-134K	0.9946	0.8574	0.8508	0.8582	0.8456
Resnet50	Phish360	VanNL126K	0.9822	0.9109	0.9786	0.9114	0.9417
Densenet121	Phish360	VanNL126K	0.9957	0.9309	0.9761	0.9388	0.9553
Resnet50	Phish360	PhishIntention	0.9918	0.9666	0.9630	0.9807	0.9701
Densenet121	Phish360	PhishIntention	0.9887	0.9615	0.9534	0.9824	0.9662
Resnet50	Phish360	PWD2016	0.9701	0.9175	0.9789	0.8414	0.8947
Densenet121	Phish360	PWD2016	0.9780	0.9333	0.9650	0.8893	0.9164

Table 6.10 Comparison between our best performing configuration of CrossPhire and existing baseline approaches employing the same benchmark datasets

Research - Dataset	Approach	Test Acc.	Test Precision	Test Recall	Test F1
Sánchez-Paniagua et al. [30] - PILWD-134K	LightGBM	0.9795	0.9830	0.9760	0.9800
Our Study - PILWD-134K	CP-EX-1	**0.9812**	**0.9849**	**0.9763**	**0.9806**
Sánchez-Paniagua et al. [26] - PWD2016	LightGBM	0.9760	-	-	-
Sánchez-Paniagua et al. [106] - PWD2016	KNN	0.9735	-	-	-
Our Study - PWD2016	CP-EX-2	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Liu et al. [29] - PhishIntention	Vision-based approach	-	0.9980	0.9060	-
Our Study - PhishIntention	CP-EX-2	**0.9963**	0.9954	**0.9983**	**0.9968**
Van Dooremaal et al. [31] - VanNL126k	SSIM	0.9966	0.9955	-	0.9977
Our Study - VanNL126k	CP-EX-2	0.9942	**0.9957**	**0.9972**	0.9964

Upon examining the related works, we observed that the employed multimodal datasets are underutilized in anti-phishing research due to various reasons that we previously mentioned, including the lack of a standard dataset and the inconsistencies within the datasets. Furthermore, since some of the datasets we used in this thesis were published recently in 2022 (i.e., PILWD-134k and PhishIntention), researchers may be still unaware of their existence or are currently working towards publishing studies utilizing them.

Given the lack of multimodal anti-phishing approaches, we searched for anti-phishing solutions that utilize the benchmark datasets. We observed that numerous studies combine self-collected data samples with sampled data from public datasets. Since comparing CrossPhire with such approaches would be inequitable, we opted to select studies that exclusively utilized one of the benchmark datasets. These studies include [26, 29–31, 106]. The approach proposed by Sánchez-Paniagua et al. [30] leverages 54 hand-crafted features extracted from the URL, HTML, and web technology. The web technology features include binary features such as Google Analytics, Apache, and Bootstrap. The authors collected and published the PILWD-134k dataset to evaluate their approach using the LightGBM classifier.

PhishIntention [29] extracts the targeted brand visually and the web page's credential-taking intention by combining several deep learning models. To locate the logos and the input box, it first locates the salient rectangular regions from the images. The logo's brand is detected using an OCR-aided recognition approach based on siamese matching. PhishIntention interacts with the webpage to confirm the credential-taking intention. The authors evaluated their proposed scheme using their collected PhishIntention dataset.

Sánchez-Paniagua et al. [26] employed the PWD2016 dataset to implement several machine learning models trained on 40 NLP-based features extracted from the web page's URL, obtaining the best performance using the LightGBM classifier. Similarly, the authors in [106] implemented the KNN algorithm on the PWD2016 dataset using the same NLP-based features to showcase the performance of the phishing detection systems using various datasets collected in different years.

Van Dooremaal et al. [31] proposed a hybrid phishing detection approach utilizing text and visual features. The authors used a subset of 2,000 samples from their published dataset. The proposed approach uses reverse image search to find similar associated websites. Then, the similarity between the suspicious and associated samples is computed to classify the web page as phishing or legitimate. Table 6.10 presents existing approaches in the literature employing these benchmark datasets.

Examining table 6.10 reveals that CrossPhire outperforms all the existing approaches. Although Van Dooremaal et al. [31] achieves a higher accuracy of 99.66% compared to our approach with an accuracy of 99.42%, however, the authors sampled 2,000 samples from the dataset without mentioning their selection criteria. So their approach is validated only using 600 samples, while our approach is trained and validated on the whole dataset, containing approximately 126,000 samples.

6.4.4. Comparing CrossPhire With Multimodal CLIP in Phishing Detection

Given the CLIP model's recent popularity and impressive results, we decided to investigate its performance by fine-tuning it. In order to examine its performance, we fine-tuned the CLIP model on the benchmark datasets. Since the CLIP model takes image-caption pairs, we utilized the screenshot images and English-translated BS text from the benchmark datasets.

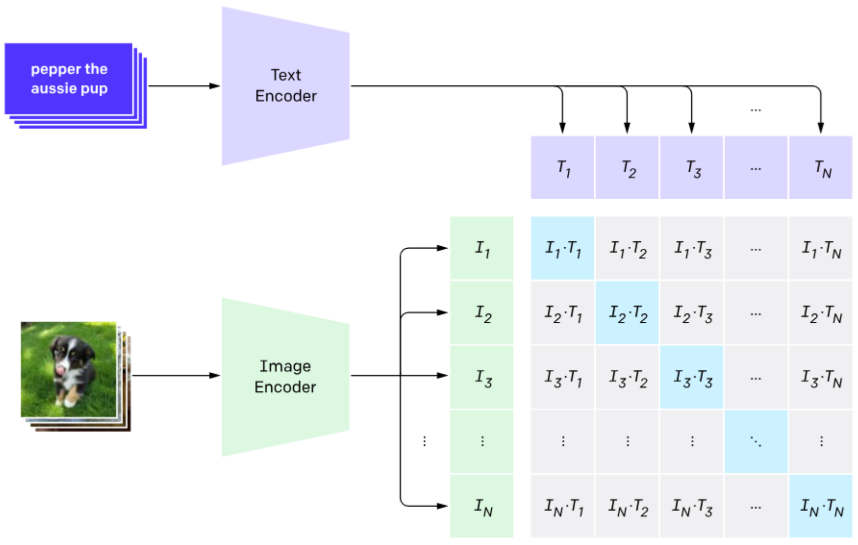

pepper the aussie pup
Text Encoder
Image Encoder
T_1 T_2 T_3 ... T_N
I_1 I_2 I_3 ⋮ I_N
$I_1 \cdot T_1$ $I_1 \cdot T_2$ $I_1 \cdot T_3$ ... $I_1 \cdot T_N$
$I_2 \cdot T_1$ $I_2 \cdot T_2$ $I_2 \cdot T_3$ ... $I_2 \cdot T_N$
$I_3 \cdot T_1$ $I_3 \cdot T_2$ $I_3 \cdot T_3$ ... $I_3 \cdot T_N$
$I_N \cdot T_1$ $I_N \cdot T_2$ $I_N \cdot T_3$ ... $I_N \cdot T_N$

Figure 6.3 Contrastive pre-training of the CLIP model (adopted from [5])

6.4.4.1. CLIP : stands for Contrastive Language-Image Pretraining, which is a groundbreaking approach proposed by Radford et al. [5]. This deep learning model, developed by OpenAI in 2021, sets itself apart from traditional visual models in the computer vision domain by employing a multimodal training approach. CLIP learns to associate images with textual descriptions (captions) in a multimodal learning framework. The authors trained the model on a huge collection of 400 million image-caption pairs that were scraped from the internet. Despite not being specifically trained on the ImageNet dataset, the model demonstrated remarkable zero-shot performance, matching the accuracy of ResNet. Given the SOTA performance achieved by CLIP, we investigated its performance by fine-tuning it on the phishing detection task.

The training process involves giving the model a batch of images and their associated textual captions, as depicted in figure 6.3. CLIP combines the text and image modalities by using the joint training of a text encoder and an image encoder, mapping their representations in a shared embedding space. The model learns to map similar image and text embedding vectors closer while pushing dissimilar pairs further. In other words, CLIP tries to minimize the distance between correct image-text pairs (maximize cosine similarity) and maximize the distance (minimize cosine similarity) between incorrect pairings. Notably, the authors

Table 6.11 Comparison between CrossPhire and fine-tuned CLIP model on phishing detection using the benchmark datasets

Model	**Dataset**	**Train Acc.**	**Test Acc.**
CLIP	Phish360	0.9938	0.9622
CrossPhire	Phish360	0.9991	**0.9921**
CLIP	PhishIntention	0.9904	0.9895
CrossPhire	PhishIntention	0.9990	**0.9963**
CLIP	PILWD-134K	0.9888	0.9648
CrossPhire	PILWD-134K	0.9904	**0.9812**
CLIP	VanNL126k	0.9842	0.9798
CrossPhire	VanNL126k	0.9995	**0.9942**
CLIP	PWD2016	0.9907	0.9922
CrossPhire	PWD2016	1.0000	**1.0000**

trained the CLIP model from scratch without initializing the image and text encoders with pre-trained weights.

Looking at figure 6.3, the process starts by encoding all the images in the batch; the encoded image vectors are shown in green (denoted by I1, I2,,, In). Similarly, the text captions are encoded using the text encoder, and the text vectors are shown in purple. The similarities of the correct image-text pairs are the diagonal values highlighted in blue. The optimization process employs a symmetric cross-entropy loss function (contrastive loss) that operates on the similarity scores.

6.4.4.2. Fine-tuning CLIP: We fine-tuned the model for phishing detection using the benchmark datasets by adding a multilayer perceptron (MLP) on top of CLIP. The MLP consists of two hidden layers having 512 neurons. We conducted joint training of both the MLP and CLIP, optimizing the network using an Adam optimizer with a learning rate of 0.0001 for 20 epochs. Our training data included screenshot images and translated body text from benchmark datasets.

We present the results obtained by the CLIP model and compare them with those achieved by CrossPhires in Table 6.11. As illustrated in the summarized results, CLIP demonstrated its highest performance on PWD2016, achieving a testing accuracy of 99.22%. However, its performance was relatively lower on Phish360, reaching 96.22%. On the other hand, CrossPhires outperformed CLIP on all datasets, achieving the highest accuracy of 100% on PWD2016 and the lowest accuracy of 98.12% on PILWD-134K.

Notably, while the CLIP model exhibited its least impressive accuracy on the Phish360 dataset, the CrossPhires model achieved 99.21%. These results show the robustness and resilience of our proposed model compared to the current state-of-the-art multimodal model, as it consistently achieved higher results on all the datasets.

7. DISCUSSION

In this thesis, we present a comprehensive solution to detect phishing web pages through an innovative end-to-end framework. Our approach extracts multimodal features from the site's URL, source code, and screenshot image to effectively identify phishing web pages. One advantage of this multimodal approach is enabling a comprehensive analysis of the web page by considering the semantical and visual aspects of the web page. Furthermore, our framework exhibits adaptability by accommodating new and emerging models. For instance, we can seamlessly integrate improved sentence transformer models capable of capturing longer text sequences for a more comprehensive understanding of textual content. Similarly, we can update the image processing component with the latest state-of-the-art models. This flexibility ensures that our solution remains up-to-date and maintains optimal performance over time.

Fine-Tuning Capabilities and Transfer Learning:

While we did not explore the fine-tuning capabilities of CrossPhire, it is important to note that our solution can be fine-tuned with new data. By leveraging a pre-trained version of GramBeddings, we can pre-train CrossPhire on the benchmarking datasets, incorporating knowledge from diverse datasets with various phishing tactics over different years. Subsequently, fine-tuning the model with new data allows it to adapt to evolving threats, ensuring consistently high performance. This adaptability stands out as one of the key advantages of deep learning networks over traditional machine learning models. In contrast to how we fine-tune pre-trained image models using screenshot images, we take a different approach with pre-trained sentence transformers. Here, we employ them as encoders, processing textual content on the fly. This means integrating the model as frozen layers, where the model weights are not updated, thereby reducing the computational demands for textual feature representation.

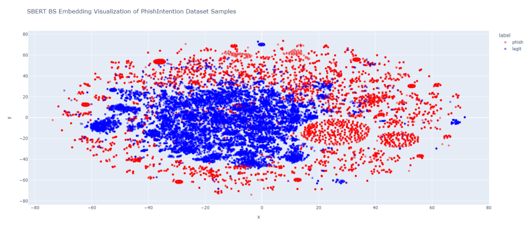
Comparative Studies:

In the dataset-wise comparative studies, we made sure to select the studies utilizing the exact datasets that we evaluated CrossPhire on. In contrast, other studies seem to compare their proposed approaches with unrelated and different approaches making inadequate comparisons. In dataset-wise comparisons, CrossPhire outperformed the baseline approaches by a notable margin. While the increase in accuracy is significant, our approach surpasses baseline methods by leveraging both visual and textual features of websites to determine their legitimacy. This advantage is achieved without relying on third-party features or requiring an extensive manual feature engineering process.

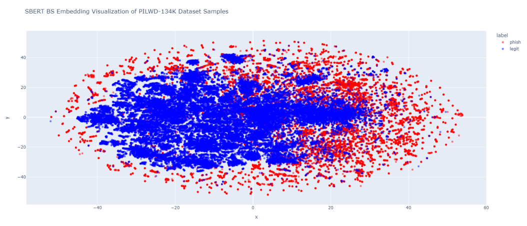
In method-wise comparisons with the bimodal CLIP model, which employs images and text, we fine-tuned CLIP using benchmark datasets to facilitate a fair assessment against CrossPhire. The performance evaluation on benchmark datasets demonstrated that CrossPhire outperformed CLIP on all the datasets, with a particularly significant margin on the Phish360 dataset. This highlights CrossPhire's robustness and effectiveness, establishing it as a highly effective anti-phishing solution across various scales and datasets.

7.1. Visualizing The Textual Content's Embeddings

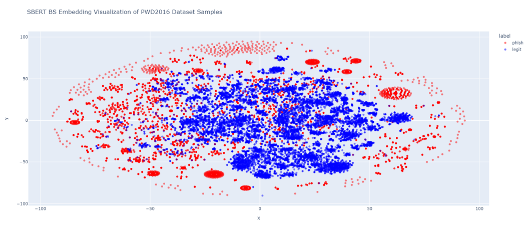
The notably high results, in line with our hypothesis, indicate that the main textual content of the web pages offered a comprehensive representation of the websites. Despite the limitation of pre-trained sentence transformers, which only consider the first 512 tokens, the resulting 768-dimensional encoded vectors effectively distinguished between the text in legitimate and phishing websites. In this high-dimensional space, the encoded textual vectors belonging to the phishing class are mapped closer together and further away from the legitimate ones. To visualize the high-dimensional vector embeddings, we utilized t-Distributed Stochastic Neighbor Embedding (t-SNE) [107], a dimensionality reduction technique that converts high-dimensional data into a two-dimensional space for easier visualization. Given that the embeddings generated by MPNet on the English-translated BeautifulSoup text provided the highest accuracy (refer to Textual Content Experiments),



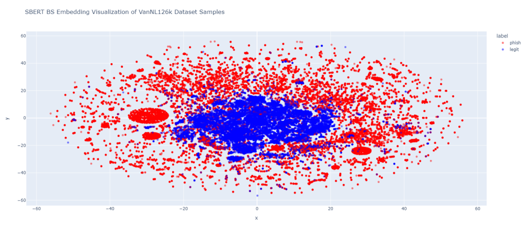
(a)



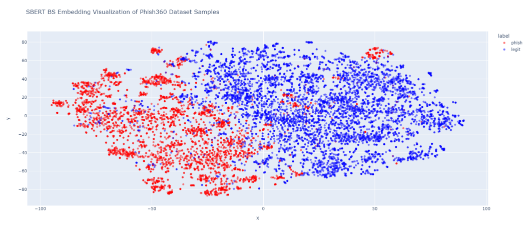
(b)



(c)



(d)



(e)

Figure 7.1 T-SNE visualization of embeddings generated with SBERT using translated_BS_text for (a) PhishIntention, (b) PILWD-134K, (c) PWD2016, (d) VanNL126k, and (e) Phish360. Phishing instances are depicted in red, while legitimate instances are depicted in blue.

we present their embedding visualizations for all datasets in figure 7.1. The sentence transformer paired with The English-translated BeautifulSoup text was able to differentiate between phishing (red points) and legitimate (blue points) instances, as demonstrated in figure 7.1. The encoded embeddings showed varying effectiveness in separating the phishing and legitimate classes across different datasets. Notably, in figure 7.1(c) representing the Phish360 dataset, the encoded vectors appear to be linearly separable, successfully capturing distinctive characteristics.

7.2. Limitations

Despite demonstrating high performance, our proposed framework has an apparent limitation which is the requirement for high computational resources. We did not explicitly mention training time due to the use of multiple computers for numerous experiments. Training deep neural networks efficiently with large multimedia datasets necessitates highly computational-powered GPUs, posing a challenge for future research. However, given recent technological advancements, acquiring such resources is becoming increasingly feasible. Moreover, developing a sustainable and robust solution remains imperative given the significant potential harm caused by phishing attacks.

8. CONCLUSION

In this thesis, we propose a novel end-to-end deep learning model named CrossPhire, designed to combat phishing attacks' increasing frequency and sophistication, particularly zero-day attacks. CrossPhire leverages contextualized, semantic, and visual features extracted from URLs, HTML source code, and screenshots of web pages to effectively differentiate between phishing and legitimate websites. Unlike existing anti-phishing solutions, CrossPhire is language-agnostic and operates independently of third-party services and manually extracted features, thereby reducing latency and vulnerabilities to zero-day attacks.

The contributions of this thesis are significant and multifaceted: (1) We introduce a novel multimodal architecture, CrossPhire, capable of capturing complex temporal, visual, and semantical relationships, thereby enhancing the robustness of phishing detection systems. (2) The utilization of state-of-the-art sentence transformers and convolutional neural networks in CrossPhire, fine-tuned for the website phishing detection task, demonstrates its effectiveness in capturing nuanced differences between phishing and legitimate web pages. (3) Through extensive data analysis and collection process, this thesis presents the Phish360 dataset, a highly diverse collection of real-world legitimate and phishing examples in numerous languages, addressing the scarcity of high-quality multimedia datasets and publishing a rich resource for future research in the anti-phishing field.

We conduct comprehensive experiments to evaluate CrossPhire's design in addition to in-data and cross-data validation using five different datasets to measure the robustness of the proposed model. Our findings indicate that CrossPhire outperforms its baseline approaches, achieving 99.21% accuracy on the Phish360 dataset and an average accuracy of 99.26% on the other four benchmark datasets. By sharing the CrossPhire codebase and supplementary materials, including the Phish360 dataset, this thesis facilitates benchmarking and enables fair comparison for future studies in website phishing detection. Overall, the findings of this thesis underscore the importance of adopting multimodal approaches, such as CrossPhire, to address the evolving landscape of cybercrimes. The research not only advances the understanding of multimodal phishing detection techniques but also opens new avenues for further exploration, particularly in optimizing and applying deep learning models for cybersecurity. Future work may explore the adaptation and scalability of CrossPhire in real-world settings and its integration into existing cybersecurity frameworks to enhance their resilience against phishing threats. By combining advanced deep learning models with rigorous evaluation methodologies using diverse datasets, this research makes significant strides toward enhancing the resilience of anti-phishing solutions in protecting individuals and entities against malicious online activities.

REFERENCES

[1] Statista. Cybercrime: Monetary damage united states 2022 — statista. `https://www.statista.com/statistics/267132/total-damage-caused-by-by-cybercrime-in-the-us/`, **2022**. [Accessed 15-01-2024].

[2] Ahmet Selman Bozkir, Firat Coskun Dalgic, and Murat Aydos. Grambeddings: a new neural network for url based identification of phishing web pages through n-gram embeddings. *Computers & Security*, 124:102964, **2023**.

[3] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708. **2017**.

[4] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, **2019**.

[5] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, **2021**.

[6] Hannah Ritchie, Edouard Mathieu, Max Roser, and Esteban Ortiz-Ospina. Internet. *Our World in Data*, **2023**. Https://ourworldindata.org/internet.

[7] Elmer EH Lastdrager. Achieving a consensual definition of phishing based on a systematic review of the literature. *Crime Science*, 3(1):1–10, **2014**.

[8] Kang Leng Chiew, Kelvin Sheng Chek Yong, and Choon Lin Tan. A survey of phishing attacks: Their types, vectors and technical approaches. *Expert Systems with Applications*, 106:1–20, **2018**.

[9] CISCO. Cybersecurity threat trends. https://umbrella.cisco.com/info/2021-cyber-security-threat-trends-phishing-crypto-top-the **2021**. [Accessed 11-01-2024].

[10] FBI Internet Crime Complain Center. Internet crime report. https://www.ic3.gov/Media/PDF/AnnualReport/2022_IC3Report.pdf, **2022**. [Accessed 15-01-2024].

[11] FBI Internet Crime Complain Center. Internet crime report. https://www.ic3.gov/Media/PDF/AnnualReport/2021_IC3Report.pdf, **2021**. [Accessed 15-01-2024].

[12] Minal Chawla and Siddarth Singh Chouhan. A survey of phishing attack techniques. *International Journal of Computer Applications*, 93(3), **2014**.

[13] Brij B Gupta, Krishna Yadav, Imran Razzak, Konstantinos Psannis, Arcangelo Castiglione, and Xiaojun Chang. A novel approach for phishing urls detection using lexical based machine learning in a real-time environment. *Computer Communications*, 175:47–57, **2021**.

[14] Anti-Phishing Working Group. Phishing activity trends reports. https://apwg.org/trendsreports/, **2023**. [Accessed 11-01-2024].

[15] Ankit Kumar Jain and BB Gupta. A survey of phishing attack techniques, defence mechanisms and open research challenges. *Enterprise Information Systems*, 16(4):527–565, **2022**.

[16] Asadullah Safi and Satwinder Singh. A systematic literature review on phishing website detection techniques. *Journal of King Saud University-Computer and Information Sciences*, **2023**.

[17] Huajun Huang, Junshan Tan, and Lingxi Liu. Countermeasure techniques for deceptive phishing attack. In *2009 International Conference on New Trends in Information and Service Science*, pages 636–641. IEEE, **2009**.

[18] Yun Lin, Ruofan Liu, Dinil Mon Divakaran, Jun Yang Ng, Qing Zhou Chan, Yiwen Lu, Yuxuan Si, Fan Zhang, and Jin Song Dong. Phishpedia: A hybrid deep learning based approach to visually identify phishing webpages. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 3793–3810. **2021**.

[19] Ahmet Selman Bozkir and Murat Aydos. Logosense: A companion hog based logo detection scheme for phishing web page and e-mail brand recognition. *Computers & Security*, 95:101855, **2020**.

[20] Thanathorn Phoka and Phisetphong Suthaphan. Image based phishing detection using transfer learning. In *2019 11th International Conference on Knowledge and Smart Technology (KST)*, pages 232–237. IEEE, **2019**.

[21] Linshu Ouyang and Yongzheng Zhang. Phishing web page detection with html-level graph neural network. In *2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 952–958. IEEE, **2021**.

[22] Tristan Bilot, Grégoire Geis, and Badis Hammi. Phishgnn: A phishing website detection framework using graph neural networks. In *Proceedings of the 19th International Conference on Security and Cryptography*, volume 1. **2022**.

[23] Mehmet Korkmaz, Ozgur Koray Sahingoz, and Banu Diri. Detection of phishing websites by using machine learning-based url analysis. In *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pages 1–7. IEEE, **2020**.

[24] Andrei Butnaru, Alexios Mylonas, and Nikolaos Pitropakis. Towards lightweight url-based phishing detection. *Future internet*, 13(6):154, **2021**.

[25] Routhu Srinivasa Rao, Tatti Vaishnavi, and Alwyn Roshan Pais. Catchphish: detection of phishing websites by inspecting urls. *Journal of Ambient Intelligence and Humanized Computing*, 11:813–825, **2020**.

[26] Manuel Sánchez-Paniagua, Eduardo Fidalgo Fernández, Enrique Alegre, Wesam Al-Nabki, and Victor Gonzalez-Castro. Phishing url detection: A real-case scenario through login urls. *IEEE Access*, 10:42949–42960, **2022**.

[27] Chidimma Opara, Yingke Chen, and Bo Wei. Look before you leap: Detecting phishing web pages by exploiting raw url and html characteristics. *Expert Systems with Applications*, 236:121183, **2024**.

[28] Mehmet Korkmaz, Emre Kocyigit, Ozgur Koray Sahingoz, and Banu Diri. Deep neural network based phishing classification on a high-risk url dataset. In *International Conference on Soft Computing and Pattern Recognition*, pages 648–657. Springer, **2020**.

[29] Ruofan Liu, Yun Lin, Xianglin Yang, Siang Hwee Ng, Dinil Mon Divakaran, and Jin Song Dong. Inferring phishing intention via webpage appearance and dynamics: A deep vision based approach. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 1633–1650. **2022**.

[30] Manuel Sánchez-Paniagua, Eduardo Fidalgo, Enrique Alegre, and Rocío Alaiz-Rodríguez. Phishing websites detection using a novel multipurpose dataset and web technologies features. *Expert Systems with Applications*, 207:118010, **2022**.

[31] Bram Van Dooremaal, Pavlo Burda, Luca Allodi, and Nicola Zannone. Combining text and visual features to improve the identification of cloned webpages for early phishing detection. In *Proceedings of the 16th International Conference on Availability, Reliability and Security*, pages 1–10. **2021**.

[32] Kang Leng Chiew, Ee Hung Chang, C Lin Tan, Johari Abdullah, and Kelvin Sheng Chek Yong. Building standard offline anti-phishing dataset for benchmarking. *International Journal of Engineering & Technology*, 7(4.31):7–14, **2018**.

[33] Brij B Gupta, Nalin AG Arachchilage, and Kostas E Psannis. Defending against phishing attacks: taxonomy of methods, current issues and future directions. *Telecommunication Systems*, 67:247–267, **2018**.

[34] Gunter Ollmann. The phishing guide-understanding and preventing phishing attacks, septembre 2004.

[35] Deanna D Caputo, Shari Lawrence Pfleeger, Jesse D Freeman, and M Eric Johnson. Going spear phishing: Exploring embedded training and awareness. *IEEE security & privacy*, 12(1):28–38, **2013**.

[36] Katharina Krombholz, Heidelinde Hobel, Markus Huber, and Edgar Weippl. Advanced social engineering attacks. *Journal of Information Security and applications*, 22:113–122, **2015**.

[37] Tzipora Halevi, Nasir Memon, and Oded Nov. Spear-phishing in the wild: A real-world study of personality, phishing self-efficacy and vulnerability to spear-phishing attacks. *Phishing Self-Efficacy and Vulnerability to Spear-Phishing Attacks (January 2, 2015)*, **2015**.

[38] Tom N Jagatic, Nathaniel A Johnson, Markus Jakobsson, and Filippo Menczer. Social phishing. *Communications of the ACM*, 50(10):94–100, **2007**.

[39] Markus Huber, Martin Mulazzani, Edgar Weippl, Gerhard Kitzler, and Sigrun Goluch. Friend-in-the-middle attacks: Exploiting social networking sites for spam. *IEEE Internet Computing*, 15(3):28–34, **2011**.

[40] Yalin Baştanlar and Mustafa Özuysal. Introduction to machine learning. *miRNomics: MicroRNA biology and computational analysis*, pages 105–128, **2014**.

[41] Issam El Naqa and Martin J Murphy. *What is machine learning?* Springer, **2015**.

[42] Yagang Zhang. *New Advances in Machine Learning*. IntechOpen, Rijeka, **2010**. doi:10.5772/225.

[43] Prakash M Nadkarni, Lucila Ohno-Machado, and Wendy W Chapman. Natural language processing: an introduction. *Journal of the American Medical Informatics Association*, 18(5):544–551, **2011**.

[44] Elizabeth D Liddy. Natural language processing. **2001**.

[45] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, **2015**.

[46] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, and Eftychios Protopapadakis. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018, **2018**.

[47] Lisa Torrey and Jude Shavlik. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI global, **2010**.

[48] Weiping Wang, Feng Zhang, Xi Luo, and Shigeng Zhang. Pdrcnn: Precise phishing detection with recurrent convolutional neural networks. *Security and Communication Networks*, 2019:1–15, **2019**.

[49] Sajjad Jalil, Muhammad Usman, and Alvis Fong. Highly accurate phishing url detection based on machine learning. *Journal of Ambient Intelligence and Humanized Computing*, 14(7):9233–9251, **2023**.

[50] Mehmet Korkmaz, Emre Kocyigit, Ozgur Koray Sahingoz, and Banu Diri. Phishing web page detection using n-gram features extracted from urls. In *2021 3rd International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, pages 1–6. IEEE, **2021**.

[51] KS Jishnu and B Arthi. Enhanced phishing url detection using leveraging bert with additional url feature extraction. In *2023 5th International Conference*

on Inventive Research in Computing Applications (ICIRCA), pages 1745–1750. IEEE, **2023**.

[52] KS Jishnu and B Arthi. Phishing url detection by leveraging roberta for feature extraction and lstm for classification. In *2023 Second International Conference on Augmented Intelligence and Sustainable Systems (ICAISS)*, pages 972–977. IEEE, **2023**.

[53] Siddharth Kumar. Malicious and benign urls. `https://www.kaggle.com/datasets/siddharthkumar25/malicious-and-benign-urls`, **2019**. [Accessed 14-11-2023].

[54] Phishtank — join the fight against phishing. (n.d.-c). `https://phishtank.com/`, **2006**. [Accessed 04-11-2023].

[55] Samuel Marchal, Jérôme François, Radu State, and Thomas Engel. Phishstorm: Detecting phishing with streaming analytics. *IEEE Transactions on Network and Service Management*, 11(4):458–471, **2014**.

[56] Dmoz. (n.d.). the directory of the web. `http://www.dmoz.org/`. [Accessed 04-11-2023].

[57] Katherine Haynes, Hossein Shirazi, and Indrakshi Ray. Lightweight url-based phishing detection using natural language processing transformers for mobile devices. *Procedia Computer Science*, 191:127–134, **2021**.

[58] Common crawl - open repository of web crawl data. `https://commoncrawl.org/`, **2007**. [Accessed 04-11-2023].

[59] Lázaro Bustio-Martínez, Miguel A Álvarez-Carmona, Vitali Herrera-Semenets, Claudia Feregrino-Uribe, and René Cumplido. A lightweight data representation for phishing urls detection in iot environments. *Information Sciences*, 603:42–59, **2022**.

[60] Mehmet Korkmaz, Ozgur Koray Sahingoz, and Banu Diri. Feature selections for the classification of webpages to detect phishing attacks: a survey. In *2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, pages 1–9. IEEE, **2020**.

[61] Ozgur Koray Sahingoz, Ebubekir Buber, Onder Demir, and Banu Diri. Machine learning based phishing detection from urls. *Expert Systems with Applications*, 117:345–357, **2019**.

[62] MANU Siddhartha. Malicious urls dataset. `https://www.kaggle.com/datasets/sid321axn/malicious-urls-dataset`, **2021**. [Accessed 27-11-2023].

[63] J Akshaya. Phishing websites detection. `https://www.kaggle.com/datasets/akshaya1508/phishing-websites-detection`, **2020**. [Accessed 27-11-2023].

[64] Mohammad Saiful Islam Mamun, Mohammad Ahmad Rathore, Arash Habibi Lashkari, Natalia Stakhanova, and Ali A Ghorbani. Detecting malicious urls using lexical analysis. In *Network and System Security: 10th International Conference, NSS 2016, Taipei, Taiwan, September 28-30, 2016, Proceedings 10*, pages 467–482. Springer, **2016**.

[65] Eduardo Benavides-Astudillo, Walter Fuertes, Sandra Sanchez-Gordon, Daniel Nuñez-Agurto, and Germán Rodríguez-Galán. A phishing-attack-detection model using natural language processing and deep learning. *Applied Sciences*, 13(9):5275, **2023**.

[66] Chidimma Opara, Bo Wei, and Yingke Chen. Htmlphish: Enabling phishing web page detection by applying deep learning techniques on html analysis. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, **2020**.

[67] Subhash Ariyadasa, Shantha Fernando, and Subha Fernando. Combining long-term recurrent convolutional and graph convolutional networks to detect phishing sites using url and html. *IEEE Access*, 10:82355–82375, **2022**.

[68] Routhu Srinivasa Rao, Amey Umarekar, and Alwyn Roshan Pais. Application of word embedding and machine learning in detecting phishing websites. *Telecommunication Systems*, pages 1–13, **2022**.

[69] Subhash Ariyadasa, Subha Fernando, and Shantha Fernando. Detecting phishing attacks using a combined model of lstm and cnn. *Int. J. Adv. Appl. Sci*, 7(7):56–67, **2020**.

[70] Jian Feng, Lianyang Zou, Ou Ye, and Jingzhou Han. Web2vec: Phishing webpage detection method based on multidimensional features driven by deep learning. *IEEE Access*, 8:221214–221224, **2020**.

[71] Subha Fernando Subhash Ariyadasa, Shantha Fernando. Phishing websites dataset. `https://data.mendeley.com/datasets/n96ncsr5g4/1`, **2021**. [Accessed 16-11-2023].

[72] Chidimma Opara. The data (look before you leap). `https://www.kaggle.com/datasets/guchiopara/look-before-you-leap`, **2022**. [Accessed 30-11-2023].

[73] Hossein Shirazi, Shashika R Muramudalige, Indrakshi Ray, and Anura P Jayasumana. Improved phishing detection algorithms using adversarial autoencoder synthesized data. In *2020 ieee 45th conference on local computer networks (lcn)*, pages 24–32. IEEE, **2020**.

[74] Routhu Srinivasa Rao and Alwyn Roshan Pais. Jail-phish: An improved search engine based phishing detection system. *Computers & Security*, 83:246–267, **2019**.

[75] Shuaicong Yu, Changqing An, Tao Yu, Ziyi Zhao, Tianshu Li, and Jilong Wang. Phishing detection based on multi-feature neural network. In *2022 IEEE International Performance, Computing, and Communications Conference (IPCCC)*, pages 73–79. IEEE, **2022**.

[76] Shih-Chun Lin, Pang-Cheng Wl, Hong-Yen Chen, Tomohiro Morikawa, Takeshi Takahashi, and Tsung-Nan Lin. Senseinput: An image-based sensitive input detection scheme for phishing website detection. In *ICC 2022-IEEE International Conference on Communications*, pages 4180–4186. IEEE, **2022**.

[77] JaidedAI. Github: Easyocr. `https://github.com/JaidedAI/EasyOCR`, **2023**. [Accessed 28-11-2023].

[78] Colin Choon Lin Tan, Kang Leng Chiew, Kelvin SC Yong, Yakub Sebastian, Joel Chia Ming Than, and Wei King Tiong. Hybrid phishing detection using joint visual and textual identity. *Expert systems with applications*, 220:119723, **2023**.

[79] Choon Lin Tan, Kang Leng Chiew, KokSheik Wong, et al. Phishwho: Phishing webpage detection via identity keywords extraction and target domain name finder. *Decision Support Systems*, 88:18–27, **2016**.

[80] Shengli Zhou, Linqi Ruan, Qingyang Xu, and Mincheng Chen. Multimodal fraudulent website identification method based on heterogeneous model ensemble. *China Communications*, 20(5):263–274, **2023**.

[81] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, **2009**.

[82] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European*

Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13, pages 740–755. Springer, **2014**.

[83] Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*, **2015**.

[84] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27. **2015**.

[85] Subhash Ariyadasa, Shantha Fernando, and Subha Fernando. Phishrepo: A seamless collection of phishing data to fill a research gap in the phishing domain. *International Journal of Advanced Computer Science and Applications*, 13(5), **2022**.

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

[87] Apache org. Apache parquet documentation. `https://parquet.apache.org/`, **2023**. [Accessed 30-12-2023].

[88] Adrien Barbaresi. Trafilatura: A web scraping library and command-line tool for text discovery and extraction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 122–131. **2021**.

[89] Leonard Richardson. Beautiful soup documentation — pypi.org. `https://pypi.org/project/beautifulsoup4/`, **2023**. [Accessed 05-10-2023].

[90] Aaron Swartz. html2text documentation — pypi.org. `https://pypi.org/project/html2text/`, **2020**. [Accessed 05-10-2023].

[91] xml dev team. lxml documentation — pypi.org. `https://pypi.org/project/lxml/`, **2023**. [Accessed 05-10-2023].

[92] Konstantin Lopukhin. htttml-text documentation — pypi.org. `https://pypi.org/project/html-text/`, **2020**. [Accessed 05-10-2023].

[93] Ankit Kumar Jain and Brij B Gupta. Two-level authentication approach to protect from phishing attacks in real time. *Journal of Ambient Intelligence and Humanized Computing*, 9(6):1783–1796, **2018**.

[94] Daniel Olszewski, Allison Lu, Carson Stillman, Kevin Warren, Cole Kitroser, Alejandro Pascual, Divyajyoti Ukirde, Kevin Butler, and Patrick Traynor. " get in researchers; we're measuring reproducibility": A reproducibility study of machine learning papers in tier 1 security conferences. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 3433–3459. **2023**.

[95] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778. **2016**.

[96] 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**.

[97] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*, **2019**.

[98] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mpnet: Masked and permuted pre-training for language understanding. *Advances in neural information processing systems*, 33:16857–16867, **2020**.

[99] Kai Ming Ting. *Confusion Matrix*, pages 260–260. Springer US, Boston, MA, **2017**. ISBN 978-1-4899-7687-1. doi:10.1007/978-1-4899-7687-1_50.

[100] Brendan Juba and Hai S Le. Precision-recall versus accuracy and the role of large data sets. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4039–4048. **2019**.

[101] David MW Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061*, **2020**.

[102] Kibreab Adane and Berhanu Beyene. Machine learning and deep learning based phishing websites detection: The current gaps and next directions. *Review of Computer Engineering Research*, 9(1):13–29, **2022**.

[103] Hung Le, Quang Pham, Doyen Sahoo, and Steven CH Hoi. Urlnet: Learning a url representation with deep learning for malicious url detection. *arXiv preprint arXiv:1802.03162*, **2018**.

[104] Pranav Maneriker, Jack W Stokes, Edir Garcia Lazo, Diana Carutasu, Farid Tajaddodianfar, and Arun Gururajan. Urltran: Improving phishing url detection using transformers. In *MILCOM 2021-2021 IEEE Military Communications Conference (MILCOM)*, pages 197–204. IEEE, **2021**.

[105] RJ van Geest, G Cascavilla, J Hulstijn, and N Zannone. The applicability of a hybrid framework for automated phishing detection. *Computers & Security*, 139:103736, **2024**.

[106] Manuel Sánchez-Paniagua, Eduardo Fidalgo, Víctor González-Castro, and Enrique Alegre. Impact of current phishing strategies in machine learning

models for phishing detection. In *13th International Conference on Computational Intelligence in Security for Information Systems (CISIS 2020) 12*, pages 87–96. Springer, **2021**.

[107] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), **2008**.

[108] SETSCI. Cognitive models and artificial intelligence conference. `https://ai-conf.com/bmyz2023/BMYZ2023_Proceedings.pdf`, **2023**. [Accessed 16-01-2024].

[109] Ahmad H. A. Almakhamreh and A. Selman Bozkır. Conference presentation. `https://docs.google.com/presentation/d/1Hj4i1uxgh7vBKka6I2Fu-ptup9YBMiWm/edit?usp=sharing&ouid=112103976447992651512&rtpof=true&sd=true`, **2023**.