

# ROBUST PHISHING URL DETECTION THROUGH DEEP LEARNING AND DOMAIN SHIFT MITIGATION

A thesis submitted in fulfilment of the requirements for the  
degree of Doctor of Philosophy

School of Computer Science  
Faculty of Engineering  
The University of Sydney

Fariza Rashid

Networked Systems and Security (NSS) Research Lab

June 2025

This research reported in this thesis was supported by the award of an Australian Government Research Training Program (RTP) scholarship, a Postgraduate Research Scholarship in Cyber Security Research and Engagement and the Defence Innovation Network Postgraduate Research Supplementary Scholarship to the PhD Candidate.



# Statement of Originality

I, Fariza Rashid, declare that this thesis titled '*Robust Phishing URL Detection Through Deep Learning and Domain Shift Mitigation*' and the content presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- This thesis has not been submitted for any other degree or purpose.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.
- No content generated by generative AI tools has been used in the preparation of this thesis.
- I certify that the intellectual content of this thesis is the product of my own work, and that all assistance received in preparing this thesis and all sources have been acknowledged.

Signed: \_\_\_\_\_

Date: \_\_\_\_\_

# Abstract

Phishing attacks are a prevailing cyber security threat, using deception to steal sensitive information from users and cause significant financial damage. Traditional detection methods such as blacklists and feature-based machine learning models struggle to keep up with rapidly evolving phishing strategies. Additionally, many existing models fail to generalise across diverse phishing URLs from different sources due to domain shifts in feature distributions, resulting in high false positive rates and reduced detection accuracy. Furthermore it is difficult to curate a well-representative training dataset as threat intelligence sharing is often restricted by privacy regulations and licensing. To address these challenges we propose a three-stage approach. First, we identify domain shift as a key barrier towards generalisable phishing URL classification methods. Using statistical tests, we analyse distributional differences across multiple datasets and find the specific features prone to distributional shifts causing reduced model transfer performance. Based on our findings we propose an unsupervised domain adaptation (UDA) framework to align source and target domains, improving generalisability without needing labels from the target data.

Second, we explore federated learning (FL) to enable collaborative model training using diverse data sources while addressing privacy concerns. We introduce FeUDA-BERT - a federated BERT-based model with a domain-aware self-attention mechanism and a self-attention-based client selection strategy that mitigates performance drop due to client data heterogeneity in feature distributions. Finally, we leverage large language models (LLMs) for phishing URL detection, harnessing LLMs' few-shot learning and reasoning abilities. We propose an explainable one-shot classification framework that achieves performance close to supervised models without large labelled datasets. Together, these stages advance phishing detection by improving generalisation, privacy, and explainability.

# Acknowledgements

I would like to express my sincere gratitude to my supervisor, Dr. Suranga Seneviratne, for granting me the opportunity to pursue my Ph.D. His constant support, insightful guidance, and encouragement throughout my candidature have made this journey both enriching and rewarding. I am especially thankful for his efforts in connecting me with researchers across the field and for continually encouraging me to seek out opportunities that have greatly contributed to my professional growth.

I also extend my heartfelt thanks to everyone who supported and collaborated with me during my Ph.D. candidature. In particular, I am deeply grateful to Ben Doyle for his invaluable guidance, and for facilitating connections to professional networks that have significantly enhanced both my academic and career development. I would also like to thank Dr. Caren Han for her support during our research collaborations. To my colleagues Thilini, Naveen, Nishavi, Bhanuka, Dishanika, and Ravin—thank you for the thoughtful discussions and your constant words of encouragement throughout this journey.

Last but certainly not least, I thank my parents, my grandparents, my siblings, my friends, and my husband for their unconditional love and unwavering support in countless ways over the past three years. This thesis would not have been possible without their belief in me. I am grateful to God for the many blessings, people, and opportunities that enabled me to complete this work.

# List of Abbreviations

AI	Artificial Intelligence
AIoT	Artificial Internet of Things
BERT	Bidirectional Encoder Representations from Transformers
CNN	Convolutional Neural Network
CTI	Cyber Threat Intelligence
DNN	Deep Neural Network
EM	Expectation Maximisation
FL	Federated Learning
IID	Independent and Identically Distributed
ISAC	Information Sharing and Analysis Centre
KL	Kullback–Leibler
LLM	Large Language Models
LSTM	Long Short-Term Memory
MLP	Multi-Layer Perceptron
RNN	Recurrent Neural Network
TF-IDF	Term Frequency - Inverse Document Frequency
UDA	Unsupervised Domain Adaptation
URL	Uniform Resource Locator

# List of Publications

## Conferences

1. **F. Rashid**, B. Doyle, S. Seneviratne, “FeUDA-BERT: Federated URL Domain-Aware BERT for Phishing URL Detection”, *IEEE Conference on Local Computer Networks (Under Review)*.
2. O. Jayawardena, **F. Rashid**, and S. Seneviratne, “Non-Contrastive Learning-based Behavioural Biometrics for Smart IoT Devices”, in Proceedings of the *19th ACM International Symposium on QoS and Security for Wireless and Mobile Networks 2023*.

## Journals

3. **F. Rashid**, N. Ranaweera, B. Doyle, S. Seneviratne, “LLMs are one-shot URL classifiers and explainers”, *Elsevier’s Computer Networks 2025*.
4. **F. Rashid**, B. Doyle, S.C. Han, S. Seneviratne, “Phishing URL detection generalisation using Unsupervised Domain Adaptation”, *Elsevier’s Computer Networks 2024*.

# Authorship Attribution Statement

We present the authorship attribution for the published works from the previous section (List of Publications) included in the thesis chapters.

- Chapter 3 of this thesis is published as [4] in the List of Publications.  
In [4], I explored feature distributional shifts in URLs, conducted statistical tests to assess the contribution of specific features in the distributional shifts, developed an unsupervised domain adaptation framework to address URL domain shift and drafted the manuscript.
- Chapter 4 of this thesis comprises content published as [1] in the List of Publications.  
In [1], I developed the concept of FeUDA-BERT, designed and conducted all experiments, and drafted the manuscript.
- Chapter 5 of this thesis comprises content published as [3] in the List of Publications.  
In [3], I developed the framework for using LLMs as one-shot classifiers and explainers for phishing detection, designed and conducted all experiments, and drafted the manuscript.

*I certify that the authorship attribution statements provided above are correct, and I have obtained permission from the other authors to include the published materials. As the lead author, and by convention in my research field, I have made the primary contribution to the publications. Additionally, I am the corresponding author for the publications included in this thesis.*

Fariza Rashid

Signature:

Date:

*As the supervisor for the candidature upon which this thesis is based, I can confirm that the authorship attribution statements above are correct.*

Dr. Suranga Seneviratne

Signature:

Date:

# Table of Contents

<b>Statement of Originality</b>	<b>iii</b>
<b>Abstract</b>	<b>iv</b>
<b>Acknowledgements</b>	<b>v</b>
<b>List of Abbreviations</b>	<b>vi</b>
<b>List of Publications</b>	<b>vii</b>
<b>Authorship Attribution Statement</b>	<b>viii</b>
<b>Table of Contents</b>	<b>ix</b>
<b>List of Figures</b>	<b>xiii</b>
<b>List of Tables</b>	<b>xv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Background . . . . .	2
1.1.1 Challenges in phishing detection . . . . .	4
1.2 Scope of the thesis: The three-stage approach . . . . .	8
1.2.1 Structure of the thesis . . . . .	9
<b>2 Related work</b>	<b>11</b>
2.1 Phishing detection methods . . . . .	11
2.2 Domain adaptation . . . . .	15
2.2.1 Background . . . . .	15
2.2.2 Fundamental work . . . . .	16
2.2.3 Application areas . . . . .	18
2.3 Federated learning . . . . .	20

2.3.1	Training process . . . . .	21
2.3.2	Application areas . . . . .	22
2.3.3	Data heterogeneity . . . . .	24
2.4	Classification using large language models . . . . .	26
2.4.1	Few-shot classification . . . . .	27
2.4.2	LLMs in cyber security . . . . .	27
<b>3</b>	<b>Unsupervised Domain Adaptation Based Detection</b>	<b>29</b>
3.1	Introduction . . . . .	29
3.2	Related work . . . . .	31
3.2.1	Approaches to phishing detection . . . . .	31
3.2.2	URL-based phishing detection . . . . .	32
3.2.2.1	Using hand-crafted features . . . . .	32
3.2.2.2	Using embedding representations . . . . .	32
3.2.3	Unsupervised Domain Adaptation . . . . .	33
3.3	Our framework . . . . .	34
3.3.1	UDA theoretical framework . . . . .	35
3.3.2	Phishing URL detection using UDA . . . . .	36
3.4	Datasets . . . . .	37
3.4.1	ISCX-2016 dataset . . . . .	37
3.4.2	EBBU-2017 dataset . . . . .	37
3.4.3	HISPAR-Phishstats dataset . . . . .	37
3.5	Transfer performance of baselines . . . . .	38
3.5.1	URLNet [1] . . . . .	39
3.5.2	URLTran [2] . . . . .	39
3.5.3	CatchPhish [3] . . . . .	39
3.5.4	PhishRF [4] . . . . .	40
3.6	Analysis of transfer performance . . . . .	41
3.6.1	New words and contradictory words . . . . .	41
3.6.2	Distributional differences in features . . . . .	44
3.7	URL feature alignment . . . . .	47
3.7.1	Changing path length . . . . .	47
3.7.2	Removing hostname dots . . . . .	50
3.8	Results . . . . .	50
3.8.1	URLNet . . . . .	51
3.8.2	URLTran . . . . .	52
3.8.3	PhishRF . . . . .	53

3.8.4	CatchPhish . . . . .	53
3.8.5	Examples for performance improvement . . . . .	55
3.9	Discussion . . . . .	55
3.9.1	Feature differences in URL datasets . . . . .	56
3.9.2	Limitations and future work . . . . .	57
3.10	Conclusion . . . . .	57
<b>4</b>	<b>FeUDA-BERT: Federated URL Domain-Aware BERT</b>	<b>59</b>
4.1	Introduction . . . . .	59
4.2	Related work . . . . .	62
4.2.1	Federated learning in cyber security . . . . .	62
4.2.2	Federated learning-based phishing detection . . . . .	63
4.2.3	Domain shift in federated learning . . . . .	64
4.3	FeUDA-BERT . . . . .	65
4.3.1	URL domain-aware self-attention . . . . .	65
4.3.2	Self attention-based client selection . . . . .	68
4.4	Evaluation . . . . .	70
4.4.1	Federated experiments . . . . .	70
4.4.2	Statistical analysis . . . . .	72
4.5	Results . . . . .	72
4.5.1	Federated experiments . . . . .	72
4.5.2	Statistical analysis . . . . .	74
4.6	Discussion . . . . .	76
4.7	Conclusion . . . . .	77
<b>5</b>	<b>LLMs are One-Shot URL Classifiers and Explainers</b>	<b>78</b>
5.1	Introduction . . . . .	78
5.2	Related Work . . . . .	81
5.2.1	Phishing URL detection . . . . .	81
5.2.2	Few-shot classification using LLMs . . . . .	82
5.2.3	Explainability of LLMs as classifiers . . . . .	84
5.3	Our framework . . . . .	84
5.4	Experiment settings . . . . .	86
5.4.1	Datasets . . . . .	87
5.4.2	Accuracy comparison with supervised URL classifiers . . . . .	87
5.4.3	Quality evaluation of LLM self-explanations . . . . .	88
5.4.3.1	Alignment between LLM and LIME indicators . . . . .	88

5.4.3.2	G-Eval framework . . . . .	90
5.5	Results . . . . .	92
5.5.1	Prediction performance of one-shot LLM URL classifiers . . . . .	92
5.5.2	Quality of LLM outputs . . . . .	94
5.5.2.1	Alignment between LLM self-explanations and LIME indicators - Jaccard similarity . . . . .	94
5.5.2.2	G-Eval analysis . . . . .	96
5.6	Extended analysis . . . . .	98
5.6.1	Zero-shot and five-shot prompting . . . . .	98
5.6.2	Increasing the training data size in the supervised setting . . . . .	100
5.6.3	Prediction consistency . . . . .	101
5.6.4	Where do LLMs perform better? . . . . .	103
5.7	Discussion . . . . .	104
5.8	Conclusion . . . . .	105
<b>6</b>	<b>Conclusion</b>	<b>106</b>
6.1	Summary and conclusion . . . . .	107
6.1.1	Phishing URL detection using Unsupervised Domain Adaptation . . . . .	107
6.1.2	Federated phishing detection with FeUDA-BERT . . . . .	108
6.1.3	One-shot phishing URL detection with large language models . . . . .	108
6.2	Future work . . . . .	109
6.2.1	Combining URL domain shift mitigation with diverse features . . . . .	109
6.2.2	Combining federated phishing detection and LLMs . . . . .	110
	<b>Bibliography</b>	<b>112</b>
<b>A</b>	<b>Appendix: One-shot LLM phishing URL classifier</b>	<b>137</b>

# List of Figures

1.1	Example of fraudulent website URL . . . . .	2
1.2	Evolution of phishing detection methods. . . . .	3
1.3	Example of domain shift in image and textual data. . . . .	5
1.4	Example of domain shift in URLs. . . . .	6
2.1	Example of concept shift. . . . .	18
2.2	General federated learning framework. . . . .	22
3.1	Phishing URL Detection using UDA. . . . .	36
3.2	Transfer performance on URLs with contradictory words . . . . .	42
3.3	Example feature distributions for PhishRF and CatchPhish. . . . .	46
3.4	Identifying overfitting features through statistical tests. . . . .	46
3.5	Examples of feature modifications. . . . .	49
3.6	Matching source path length distribution to target path length distribution . . . . .	50
3.7	Boxplot of path length distributions . . . . .	52
3.8	Boxplot of path length distributions . . . . .	53
3.9	Examples of improvement in phishing detection after feature modifications. . . . .	54
4.1	Relative importance of URL tokens based on position . . . . .	67
4.2	Target self-attention map . . . . .	68
4.3	Self-attention-based client selection. . . . .	69
4.4	F1 score results for federated experiments . . . . .	73
4.5	Accuracy results for federated experiments. . . . .	74
5.1	Example URL classification prompt and the output . . . . .	80
5.2	LLM-based one-shot URL classification framework . . . . .	86
5.3	Prompting the LLM to list benign and phishing indicators . . . . .	89
5.4	Using the G-Eval framework to assess the quality of LLM self-explanations . . . . .	91
5.5	Cumulative distribution of the Jaccard similarity scores . . . . .	95
5.6	Distribution of G-Eval scores for readability, coherence and informativeness . . . . .	97

5.7	Change in F1 scores while increasing the training set size . . . . .	100
5.8	Histograms of Gini impurity values for assessing prediction consistency . .	102

# List of Tables

3.1	Dataset summary . . . . .	38
3.2	F1 scores for baseline experiments . . . . .	40
3.3	Sign test p-values . . . . .	43
3.4	Common important features of PhishRF and CatchPhish models . . . . .	45
3.5	Mann-Whitney U test p-values . . . . .	47
3.6	Transfer performance after feature alignment . . . . .	51
4.1	PhiUSIIL Dataset Summary . . . . .	70
4.2	Wilcoxon signed-rank tests comparing baseline BERT and FeUDA-BERT . . . . .	75
4.3	Wilcoxon signed-rank tests comparing client selection methods . . . . .	76
5.1	Phishing URL detection related work summary . . . . .	83
5.2	Prediction performance of one-shot classification . . . . .	92
5.3	Percentage of Jaccard similarity scores $\geq 0.5$ . . . . .	95
5.4	Examples of alignment between LLM self-explanation and LIME indicators . . . . .	96
5.5	Prediction performance - F1 Score of zero-shot and five-shot settings . . . . .	100
5.6	Cases of similar and dissimilar predictions . . . . .	103
A.1	Prediction performance of one-shot classification . . . . .	137
A.2	Comparing LLM one-shot classification with supervised models . . . . .	138
A.3	Prediction performance of one-shot classification for PhiUSIIL . . . . .	138

# Chapter 1

## Introduction

Phishing attacks remain one of the most pervasive forms of cyber threats, exploiting both human trust and technological vulnerabilities to compromise network security. These attacks deceive users into revealing sensitive information through manipulative tactics such as urgent payment notices or fraudulent websites that mimic legitimate ones (Figure 1.1). By leveraging communication channels such as emails, text messages, and social media, phishers aim to extract confidential user credentials such as usernames, passwords, credit card details, and banking information. The stolen data enables a range of cybercrimes, including identity theft, unauthorized access to financial accounts, and large-scale breaches of data privacy.

On both individual and organizational levels, phishing results in substantial financial losses and reputational damage. According to the Australian Competition & Consumer Commission's (ACCC) Annual Report 2024, the financial losses due to phishing and related scams in Australia alone amounted to an alarming \$2.74 billion in 2023 [5]. Meanwhile, the U.S. Federal Bureau of Investigation's (FBI) Internet Crime Complaint Center (IC3) reported that total financial losses from cybercrime surpassed \$12.5 billion in 2023, marking a 22% increase from the previous year [6]. Further concerning is that IC3 recorded losses of \$1.6 billion in the first quarter of 2024 (January to May), which was \$300 million higher than the same period in 2023 [7]. Similarly, the IBM X-Force Threat Intelligence Index 2024 revealed that phishing was the most commonly used initial attack vector, accounting for 30% of cyber intrusions in 2023. This figure ties with the exploitation of valid credentials — another attack vector often facilitated by phishing [8]. The IBM report also highlighted the exponential growth in phishing attacks, attributing the increase to advancements in artificial intelligence (AI). AI-powered phishing automation now enables cybercriminals to create sophisticated phishing emails in under five minutes, a notable improvement from the average 16 hours required for manual generation.



Figure 1.1: Example of fraudulent website mimicking a legitimate Commonwealth Bank website using Cyrillic font for the letter ‘a’.

The growing sophistication and prevalence of phishing attacks necessitate the development of robust detection and mitigation strategies. We discuss the development of phishing detection methods below.

## 1.1 Background

Phishing detection methods have evolved significantly over the past few decades. In the early 2000s, web browsers relied on simple rule-based systems and manually curated blacklists to check website URLs, either blocking access to malicious sites or providing users with cautionary warnings [9]. Email providers also implemented basic heuristics, such as keyword-based filtering and sender reputation analysis, to detect suspicious messages [10]. To this day, security vendors and organizations compile lists of known phishing URLs and email addresses (e.g., VirusTotal, PhishTank). However, solely relying on such blacklists has become ineffective in the face of evolving phishing tactics.

As phishing attacks became more sophisticated, detection methods shifted toward machine learning-based approaches. These models learn statistical patterns from large annotated datasets to enable automated phishing classification. Security systems using machine learning models use email metadata, URL structures, and website content to identify phishing attempts. The different categories of features used by these models include content-based features such as webpage titles and HTML structure, and lexical features such as URL length and checking the presence of known brand names [11, 12, 13]. Heuristic-based models incorporated features such as domain age and the presence of specific keywords. Others extend their analysis to include hyperlink information, examining the relationships between internal and external links [14]. These extracted features serve as input for machine learning models such as random forests and support vector machines.

The development of deep learning models further improved phishing detection. In addition to the above features, features such as company logos and embedded images were used in convolutional neural networks (CNNs) [1]. Other sequential models, such as recurrent neural networks (RNNs) [15, 16] and long short-term memory (LSTM) networks [17], were

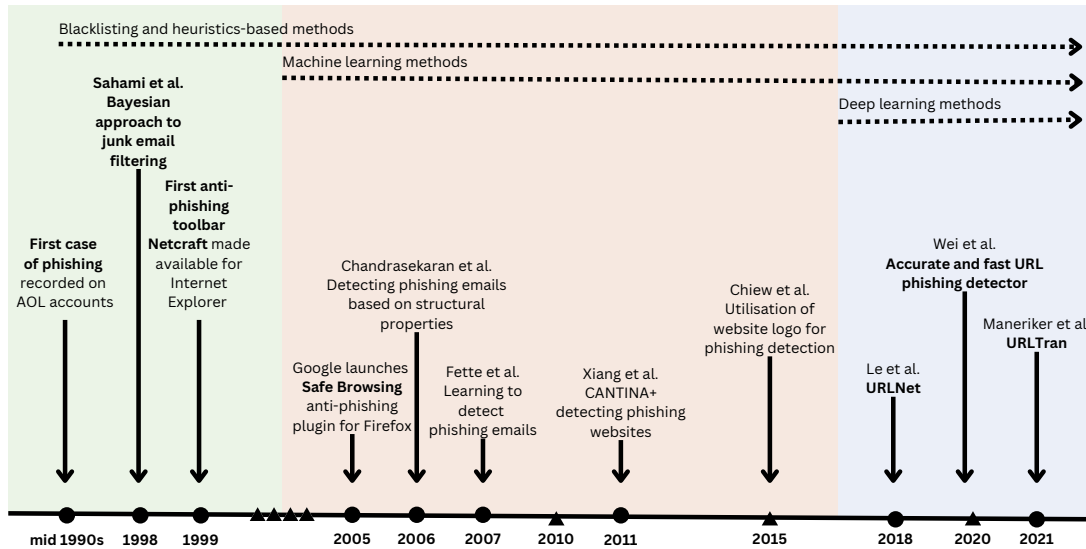


Figure 1.2: Evolution of phishing detection methods.

useful for analysing text-based features with long dependencies such as webpage URLs or HTML and Javascript content [18, 12]. Additionally, certain models incorporate third-party data sources, such as domain registration details, hosting provider records, and search engine query results related to the URL [19, 18]. Figure 1.2 displays some highly cited work to depict the evolution of phishing detection methods.

While deep learning methods have improved classification performance, practical limitations such as the short lifespan of phishing websites constrain the effectiveness of these feature-based methods. Although phishing attacks continue to grow in number, individual phishing campaigns are designed to be short-lived. This minimises their exposure to detection mechanisms and attackers capitalise on the time it takes for an attack to be detected. Research by Oest et al. [20] indicates that the time span between the first and last victim of a phishing campaign is, on average, only 21 hours. Similarly, a Kaspersky report found that one-third of phishing websites cease to exist within a single day [21].

Furthermore, phishers employ various evasion techniques to circumvent traditional detection methods. Blacklisting, for example, relies on maintaining an updated list of known phishing URLs and is hindered by the rapid turnover of phishing sites. On average, blacklisting takes 55 minutes to identify and list a phishing website, however, this delay can be extended to nearly three hours when attackers use simple evasion tactics such as URL shorteners for redirection [22]. By the time a phishing site is blacklisted, it may have already been abandoned or replaced, rendering traditional blacklisting an ineffective countermeasure against zero-day phishing attacks.

Short lifespans also restrict the availability of webpage content for analysis. That is, supervised machine learning and deep learning models depending on statistical patterns

learned from out-of-date webpage content features or third-party information will not be useful when classifying newly registered sites. Even when a phishing site remains accessible, actively visiting the webpage to extract features introduces security risks, including the potential for inadvertent malware downloads. Additionally, cybercriminals employ evasion techniques such as cloaking which applies filters to webpage content such as the IP address, geolocation, session cookies and browser fingerprint when anti-phishing crawlers access the webpage. This further complicates reliable detection.

In response to the challenges of webpage content analysis, URL-only detection methods have recently gained increasing attention as URLs are the most direct and accessible feature of a phishing site. These methods analyse URL lexical attributes, such as the presence of specific words, character sequences, and token distributions, alongside structural properties including URL length and entropy. These features are used to train both traditional machine learning classifiers, such as decision trees and random forests, and deep learning architectures, such as CNNs and recurrent neural networks (RNNs) [16, 17]. More recently, transformer-based models, such as BERT and GPT, have been adapted for phishing URL classification by leveraging contextual embeddings and sequence analysis [2]. By transforming URLs into numerical representations or embeddings, these models can capture subtle linguistic and textual URL patterns that traditional heuristics and feature-based methods might overlook.

**URL-only phishing detection.** Our primary aim in this thesis is to enhance phishing detection performance. While existing research has explored a wide range of features for detection, this thesis focuses on methods that rely solely on phishing URLs. These approaches do not depend on webpage content or external third-party sources, thereby streamlining data preprocessing steps and improving overall efficiency. Moreover, by avoiding direct access to webpage content, these methods minimize exposure to evasion techniques employed by attackers and reduce the risk of retaliatory actions. However, our exploration of existing phishing URL classifiers revealed issues that pose persistent obstacles to any phishing detection method. We elaborate on this below.

### 1.1.1 Challenges in phishing detection

#### Domain shift among phishing URLs

In machine learning, domain shift refers to changes in the distribution between the source data used to train a model and the target data encountered during evaluation or inference.

	Image Classification	Text - Sentiment Analysis
Domain 1	 <p>(photographs)</p>	<p>I bought this book to learn the piano. The music is at times hard to read because we think the book was published for singing from more than playing from. Great purchase though!</p> <p>(purchase review)</p>
Domain 2	 <p>(sketches)</p>	<p>Finnish Talentum reports its operating profit increased to EUR 20.5 mn in 2005 from EUR 9.3 mn in 2004 , and net sales totaled EUR 103.3 mn , up from EUR 96.4 mn .</p> <p>(financial review)</p>

Figure 1.3: Example of domain shift in image and textual data. Domain shift for each category is represented by the difference in feature distributions. For instance, the colour range of image domain 1 is RGB colour scale whereas the range of domain 2 is greyscale.

This shift leads to a decline in the model’s performance, as the statistical patterns and distributions learned during training may no longer be applicable when tested on the target data, which follows a different distribution.

Figure 1.3 illustrates a simple example of domain shift across two data modalities—images and text. In the image domain, the two represented categories are photographs and sketches. One notable distributional difference between these domains is colour range: photographs (domain 1) utilize an RGB colour scale, whereas sketches (domain 2) are in grayscale. Similarly, in the text domain, the figure contrasts two types of reviews: customer purchase reviews and financial reviews. A key distributional difference, in this case, is the frequency of numerical values—financial reviews (domain 2), are more likely to contain numbers, such as monetary values or stock prices, whereas purchase reviews (domain 1) primarily contain qualitative descriptions. Domain shifts are known to harm model performance and multiple studies have developed domain adaptation methods to address this [23, 24, 25].

Our work identifies the presence of domain shifts among phishing URLs. Specifically, phishing URLs collected from different sources and vantage points exhibit variations in key feature distributions. Furthermore, attackers continuously diversify URLs to evade detection. As a result, detection models trained on URLs from one source often experience a significant drop in classification performance when applied to URLs from a different source. Such domain shift concerns in other domains generally arise from varying sources of training and test data such as data collected from different communities of people or different sensors. Sometimes domain shift exists due to external factors. For instance, a COVID detection model trained during the disease outbreak is likely to be ineffective many years

Domain 1
<p><b>Benign URLs</b></p> <p><a href="https://www.facebook.com/lidl/perucanmeatloafdouclick">https://www.facebook.com/lidl/perucanmeatloafdouclick</a>  <a href="https://www.facebook.com/rugbyrama/putharchivesbukhtikanengineer">https://www.facebook.com/rugbyrama/putharchivesbukhtikanengineer</a>  <a href="https://www.facebook.com/officialplt/miscellania000016572swarovskichainfire">https://www.facebook.com/officialplt/miscellania000016572swarovskichainfire</a>  <a href="https://www.yahoo.com/irontests.wows">https://www.yahoo.com/irontests.wows</a>  <a href="https://www.yahoo.com/beauty/venuspp44201403referees">https://www.yahoo.com/beauty/venuspp44201403referees</a></p> <p><b>Phishing URLs</b></p> <p><a href="http://olx.order-complete.info/receive/59506623/invitationapostle">http://olx.order-complete.info/receive/59506623/invitationapostle</a></p>
Domain 2
<p><b>Benign URLs</b></p> <p><a href="http://olx.co.id/i2/elektronik-gadget/tablet/huawei/elektronik-gadget/tablet/huawei">http://olx.co.id/i2/elektronik-gadget/tablet/huawei/elektronik-gadget/tablet/huawei</a></p> <p><b>Phishing URLs</b></p> <p><a href="http://faceboook.bitballoon.com/">http://faceboook.bitballoon.com/</a>  <a href="http://www.facebook.com.infounknown.pealbume.com/">http://www.facebook.com.infounknown.pealbume.com/</a>  <a href="http://www.yahooka.com/index.php">http://www.yahooka.com/index.php</a></p>

Figure 1.4: Example of domain shift in URLs. URL domains of the benign URLs in Domain 1 are similar to those of the phishing URLs in Domain 2. Consequently, a model trained on domain 1 can experience a drop in performance when tested on model 2.

down the line as the disease incidence distribution is different then.

However, given the dynamic and adversarial nature of phishing attacks, domain shifts in this area are inherent and phishing detection models must be robust to this. Attackers continuously modify and diversify the characteristics of phishing URLs to evade detection systems and this underscores the need for adaptive models capable of generalizing across different sources and phishing strategies. Figure 1.4 exemplifies the changes in feature distributions due to domain shifts in URLs. We observe that the URL domains of the benign URLs in Domain 1 are similar to those of the phishing URLs in Domain 2. For example,

- <https://www.yahoo.com/irontests.wows> is benign in Domain 1 but
- <http://www.yahooka.com/index.php> is phishing in Domain 2.

Consequently, a model trained on Domain 1 may be inclined to predict the URL <http://www.yahooka.com/index.php> in Domain 2 as benign which is incorrect.

Again a key reason for vulnerability to domain shifts is the sheer volume and rapid generation of both benign and malicious sites. It is nearly impossible to curate a URL training

dataset that is perfectly representative of all existing benign hosting sites and malicious domains. Furthermore, data sharing — such as network traffic or URLs accessed within an organization — is heavily restricted by licensing agreements and privacy regulations. This limitation further hinders the ability of phishing detection models to generalise across a broader range of URLs.

### **Anomalous benign network traffic and high false positive rates**

The overwhelming volume of phishing attempts places significant strain on security analysts and leads to an increased rate of false positives in detection systems. A study surveying cybersecurity practitioners found that 99% of phishing alerts generated by detection models were false positives [26]. In other words, security teams spent valuable time and resources investigating benign incidents rather than actual threats. This inefficiency not only disrupts workflow but also contributes to a phenomenon known as alert fatigue. This is when analysts are repeatedly exposed to high volumes of false alarms, and become desensitized to warnings, increasing the risk of overlooking genuine phishing threats. Reducing false positives is therefore a crucial challenge in improving phishing detection mechanisms.

Besides the large volume of malicious traffic, the dynamic nature of legitimate internet traffic also contributes to the false positive problem. This is because many detection methods operate based on anomaly detection. They depend on knowledge of well-reputed websites and identify malicious events within network traffic by flagging URLs that deviate from known, reputable domains. This approach relies on reference lists such as the Alexa top domains list or the Tranco rankings of frequently visited domains [27]. These lists would not include newly registered legitimate websites, software license update links, or lesser-known domains. Detection methods relying on such lists would mistake these benign events as harmful.

The above challenges underscore the need for phishing detection measures that are timely, accurate, and efficient. Addressing the problem of domain shift among URL datasets requires a deep understanding of the feature distributional differences between URLs from different sources. On the other hand, reducing false positive alerts and dealing with anomalous benign network traffic requires increased knowledge of existing benign websites. In the following section, we discuss how this thesis addresses these challenges.

## 1.2 Scope of the thesis: The three-stage approach

In this thesis, we aim to develop robust phishing detection methods by addressing the challenges outlined above. We identify that the lack of generalisation in existing detection models is primarily due to domain shifts among phishing URLs. Additionally, we acknowledge other key obstacles hindering progress in this field, including high false positive rates, and vulnerability to anomalous benign traffic. Furthermore, we recognise that resistance to intelligence sharing, often due to data privacy concerns, also impedes advancements in phishing detection. To address these challenges, we propose a three-stage approach, which we describe below.

**Unsupervised Domain Adaptation.** The first stage of our work conducts an in-depth investigation into the domain shift problem in phishing URL classification. We assess the impact of domain shift using phishing URLs collected from diverse sources and demonstrate that detection models trained on one specific dataset (source dataset) fail to generalise to URLs from a different dataset (target dataset). Using three distinct phishing URL datasets, we analyse the role of specific features in model performance and examine differences in feature distributions across datasets.

To quantify the impact of these differences, we use statistical tests and establish that variations in feature distributions contribute to the observed decline in detection accuracy when models are applied to a target dataset. To address this limitation, we propose a framework based on unsupervised domain adaptation (UDA). Our UDA-based approach allows a model trained on one source dataset to perform well on a different target dataset. A key contribution of this stage is the identification of dataset-specific features that introduce noise and create domain shifts during training. Additionally, we demonstrate the performance improvements achieved through our UDA framework.

**Federated learning.** In the next stage of our work, we address the challenge of anomalous benign traffic and high false positive rates. As these occur due to the lack of access to a broader representation of website URLs, we essentially address the problem of intelligence sharing in this stage. Threat intelligence sharing—such as benign network traffic data or information about previous malicious attacks—would be particularly valuable in phishing detection, as it would expose detection models to a broader range of benign and malicious incidents during training. As a result, models would become more robust to diverse URL patterns. However, we acknowledge that strict data privacy regulations often prevent organisations from sharing such intelligence. To address this, we explore federated learning solutions, which enable multiple clients or organisations to collaboratively train a server

model without directly sharing their local data.

Since federated learning does not require the explicit sharing of raw data, it ensures compliance with data privacy laws and licensing requirements while enabling the development of a decentralised classification model that benefits from all the data available within the federated system. Our work in this stage builds on the findings from the first stage, where we analysed URL-specific features contributing to domain shift. Building on this domain knowledge, we propose a federated learning-based solution that mitigates domain shift in phishing URL detection, ultimately leading to a more robust phishing detection model.

**Large language models.** Previously, we noted that for improving detection performance, understanding existing benign sites is just as crucial as recognising phishing characteristics. The lack of proper representation of both benign and phishing URLs, as well as their associated entities, hinders the generalisation of existing detection models. One approach to addressing this issue is to leverage the knowledge embedded in large language models (LLMs). Since LLMs are trained on vast amounts of internet data which includes benign sites and entities, we explore their effectiveness in URL classification.

In this final stage of our work, we propose a one-shot classification framework that utilises LLMs' reasoning capabilities to enhance prediction accuracy. Our solution offers a lightweight approach to URL classification by reducing the need for careful dataset curation, which is typically required for supervised training. Additionally, we address the challenge of model explainability by incorporating LLMs' reasoning abilities into our framework. Our proposed solution achieves prediction performance comparable to supervised approaches that rely on large training datasets, while significantly reducing the dependency on extensive labelled data.

## 1.2.1 Structure of the thesis

The rest of this thesis is structured as follows.

**Chapter 2** provides a comprehensive review of existing literature relevant to this thesis. More specifically, we review the machine learning and deep learning concepts that are used for recent phishing detection methods. We begin by outlining the models employed for traditional heuristics-based approaches and establish the theoretical foundation of deep learning methods used for URL classification. This is followed by a theoretical explanation of domain shift to elicit why it harms machine learning and deep learning training. Next, we cover the theoretical foundations of federated learning and how this approach enables data

sharing while maintaining organisational data privacy. Finally, we cover the development of large language models (LLMs) and their use as classification models, highlighting their capabilities in reasoning and contextual knowledge and understanding.

**Chapter 3** introduces the first stage of our work. We demonstrate the existing domain shift problem among phishing URLs. We analyse URL datasets collected from various sources and investigate differences in feature distributions. Using statistical tests we identify the distributional feature differences among source and target datasets and verify that such domain shifts contribute to the drop in performance of state-of-the-art phishing URL classification models. We use an unsupervised domain adaptation (UDA) based feature alignment framework to address this issue and demonstrate that our method results in improved transfer performance. Specifically, our framework resulted in an average F1 score increase of 0.06 and as high as 0.2 in some scenarios.

**Chapter 4** describes the second stage of our work. It addresses the problem of data sharing restrictions among organisations contributing to a lack of generalisability of phishing detection models. In this work, we implement a federated learning framework to develop an improved phishing URL classification model that draws from multiple sources (clients in the federated learning framework) of URLs while maintaining the privacy of each client's URL dataset.

**Chapter 5** addresses the lack of knowledge of a broader range of benign sites and entities in existing phishing URL detection models. By leveraging the vast embedded knowledge in pre-trained large language models (LLM), we investigate their ability to classify phishing URLs and also explain predictions. Specifically, our work draws on large language models' reasoning capabilities and proposes a one-shot URL classification framework that provides explanations for each prediction based on LLM knowledge of benign hosting sites and URL patterns. We demonstrate that our one-shot framework provides prediction performances comparable to supervised models and also conduct a quantitative analysis of the prediction explanations to demonstrate alignment with post-hoc explanations of the supervised classifiers.

**Chapter 6** concludes the thesis by summarising the results and discussing the limitations and possible future directions of research.

# Chapter 2

## Related work

In this chapter, we review existing literature related to our work in this thesis. As mentioned in Chapter 1, this thesis presents a three-staged approach to phishing URL detection where we first investigate the problem of domain shift among phishing URLs, then explore the use of federated learning to alleviate the obstacles of threat intelligence sharing, and finally, we leverage the reasoning capabilities of large language models for phishing URL classification. As the main aim of the three stages is to address the problem of phishing detection, in this section we cover some related work in this area as well as studies relevant to the methods we implemented in this thesis. More specifically we discuss the following:

- **Phishing detection methods:** We discuss traditional phishing detection methods including traditional blacklisting and simple rule-based methods, as well as the more recent machine learning and deep learning methods.
- **Mitigating domain shifts:** We provide a theoretical outline of domain shifts and explain how they adversely affect the machine learning and deep learning training process. Furthermore, we provide some example studies where domain shift was addressed in the areas of image and text classification.
- **Federated learning:** We cover the theoretical basis of federated learning and discuss other studies that implemented FL for privacy-preserving model training.
- **Large language models:** We finally explore the use of large language models in existing literature with a particular focus on few-shot classification.

### 2.1 Phishing detection methods

**Blacklisting.** Phishing attacks today cover a wide range of communication channels including emails, text messages and social media platforms. As the first phishing campaigns

originated on email platforms, the original prevention methods operated on a browser level and initial examples include the anti-phishing Netcraft toolbar built for Internet Explorer. Alongside similar solutions such as the Google Safe Browsing anti-phishing plugin and Microsoft Defender Smartscreen, these solutions made use of blacklists and simple heuristics to identify suspicious webpages and give cautionary warnings to users [9].

However, primarily relying on blacklists has lost traction among phishing detection methods as these lists require frequent updates and maintenance in order to remain relevant and effective. Identification of phishing websites followed by blacklisting is further hampered by detection evasion techniques by used attacks. For example, Oest et al. [22] conducted a longitudinal study, investigating the effectiveness of blacklists over time and against various evasion techniques. The study found blacklists to become ineffective against existing techniques of cloaking and behaviour-based Javascript evasion.

**Email filters.** While blacklists operated on a browser level, email filters developed simultaneously to ensure phishing or spam emails did not reach users in the first place. The Thunderbird 1.5 email filter [28] launched in 2006, for example, used IP-based URLs, mismatched links and embedded HTML extracted from emails to alert users to potentially dangerous emails. A study by Albrecht et al. [29], conducted around the same time, proposed an open-source spam filtering framework named Spamato. This solution integrated multiple filters including URL-based filters and combined the results of all filters to produce a final decision for an email.

Fette et al. [30], on the other hand, addressed email filtering using machine learning techniques. Here, a Random Forest model was trained on email features such as IP addresses, the age of linked-to domain names, HTML information, the number of links, the number of dots in URLs, whether the email contains Javascript, and the TF-IDF of key terms. The study evaluated the proposed solution on 860 phishing emails and 6950 non-phishing emails and presented over 96% accuracy in identifying phishing emails while misclassifying legitimate emails by 0.1%.

**Content-based methods.** Other types of detection methods focused on detecting malicious websites by utilising content-based features. An early work by Provos et al. [31], proposed focusing on drive-by downloads and obfuscated JavaScript detection. This is specifically for those URLs which automatically install and run malware on users' computers. The method analysed malicious content injected into legitimate sites through iFRAMEs and found that around 1.3% of incoming search queries on Google returned at least one such URL in the results.

A similar early study focused particularly on phishing, was that of Zhang et al [32]. This work proposed the detection tool CANTINA, which used the TF-IDF information of terms on the webpage. More specifically, the tool calculated the TF-IDF scores of each term on the webpage and generated a lexical signature for five terms with the highest TF-IDF weights. This lexical signature is fed into a search engine (Google) and if the domain name of the test webpage does not match the domain name of the N top search results, the site is considered phishing. The paper evaluated this method on 200 URLs, and reported a 95% accuracy in labelling phishing websites. Similarly, Abu-Nimeh et al. [33] compared multiple machine learning models including logistic regression, support vector machines, random forests and neural networks on the TF-IDF features of over 2000 phishing and legitimate emails and demonstrated that random forests outperformed the other models with an error rate of 7.72%.

In contrast to the above, Lin et al. [34] proposed Phishpedia, a convolutional neural network-based model that extracts visual elements from a webpage and combines image and metadata analysis for phishing detection. More specifically, the model measures the similarity between visual elements from the webpage and well-reputed or familiar brands and logos to detect phishing sites. Other comparable works include [35, 36, 18].

**Detecting malicious domains.** Methods focusing particularly on malicious domains form another category of phishing detection. A significant contribution in this area is the work by De Silva et al. [37], who studied malicious hosting environments through the classification of compromised versus attacker-owned domains. This work was motivated by the fact that the mitigation strategy for a malicious website should depend on how it is hosted. A malicious site hosted on a public apex domain for example needs targeted mitigation by only being blocked at the subdomain level. Attacker-owned domains can be more decisively blacklisted. Accordingly, the study developed a Random Forest model trained on features such as domain registration patterns and historical usage, and achieved over 97% accuracy in distinguishing between sites unintentionally compromised and those deliberately created to host malicious content.

Similarly, Huang et al. [38] developed Gossip, a system that automatically identifies malicious domains by mining security-focused mailing lists. The research trained classifiers including Naïve Bayes, SVM, and Random Forests using features such as email subject lines, attachment counts, URL extensions, and domain reputation indicators. The results demonstrated that Gossip identified 79% malicious domains earlier than existing public blacklists and when tested on public mailing lists, the method showed a 94% detection accuracy. Other work in this category include [39, 40, 41].

**URL-based phishing detection.** The related work discussed above address the problem of phishing through a multitude of angles, utilising various aspects of phishing attacks as points of analysis and classification. Among these developed URL-based detection methods, URLs present the most direct access to the attack surface. Several studies have advanced this field using both traditional and deep learning techniques.

One of the earliest works on phishing URL detection using machine learning is that of Garera et al [42]. The study used features including the presence of red flag keywords, features based on Google page rank and Google web page quality guidelines, URL length, and the presence of certain structural elements like the number of dots in the domain, extracted from several million URLs. The framework incorporates these heuristic insights into training a logistic regression model, successfully flagging suspicious URLs with 97.3% accuracy over a sample of 2,500 URLs. This work helped pave the way for feature-based phishing URL detection, highlighting the predictive power of URL characteristics alone.

In another study, McGrath and Gupta [43] investigated phishing URLs by analysing WHOIS records, IP locations and geographic information. The study further analysed properties such as the differing length and character frequencies of phishing URLs and legitimate URLs and elaborated on attackers' use of free web hosting services and well-reputed brand names. Instead of proposing a specific detection algorithm, this study characterised phishing domains and uncovered behavioural trends in how attackers register, host, and deploy phishing websites, and thereby, aimed to provide heuristics based on which URL-based phishing detection methods can be further developed.

Following this, Ma et al. [44] proposed a detection method where Naïve Bayes classifiers were trained using lexical and host-based features of URLs to predict phishing. Unlike blacklists, this method relies on statistical patterns in the structure of URLs, including attributes such as hostname length, whole URL length, number of dots in the URL etc. The method also utilised host-based features such as IP-address properties, WHOIS properties, domain name properties and geographic properties. By training on labelled datasets of malicious and legitimate URLs, Ma et al. achieved high detection rates (between 95-99% accuracy).

More recently, a significant study by Mamun et al. [45] extracted a broader range of lexical features from over 100,000 URLs. Additional features in this work included the average token length with the URL path component, an entropy measurement of URL domain characters, and the ratio of domain length to URL length among others. Using a random forest classifier, the proposed solution demonstrated 97-99% detection accuracy. Another significant recent proposed CatchPhish [3] - a random forest-based classifier trained on a

combination of hostname features, URL features and TF-IDF features, demonstrating performance of 98.25% and 97.49% accuracy on benchmark datasets.

With the development of natural language processing methods, phishing detection methods began to use embedding representations of URLs. For example, URLNet [1] and URL2Vec [46] leveraged textual representations of URLs. URLNet implemented convolutional neural networks (CNNs) to automatically learn patterns from word-level and character-level embedding representations of URLs. URL2Vec, on the other hand, focused only on character-level embeddings, processing each URL component separately to train an XGBoost model. The method is evaluated on a dataset of over a million URLs and achieves an F1 score of 96.6%.

The use of URL embedding representations was further enhanced with the development of transformer models. In this area, Maneriker et al. [2], presented the first work that encoded URLs to train a BERT (Bidirectional encoder representations from transformers) classifier named URLTran. The study demonstrated that URLTrain achieved a true positive rate (TPR) of 86.80% compared to 71.20% for the next best baseline.

## 2.2 Domain adaptation

### 2.2.1 Background

As mentioned in Chapter 1, our work with URL-detection methods reveals a domain shift problem among phishing URLs. URLs collected from different sources or vantage points reflect varying feature distributions which present difficulties in the machine learning training process. In this section we expand on the concept of domain shift, its various categories and mitigation strategies.

Machine learning models traditionally assume that the training and test data originate from the same distribution. However, in practice, when discrepancies arise due to evolving statistical properties, dataset shifts over time, or variations in data collection sources, model performance can degrade significantly when a naively trained classifier is applied to unseen test data. This issue, is known as *domain shift*.

To address domain shift means to enable knowledge transfer between the two datasets or domains of differing distributions. In general, these are referred to as the *source domain* (a well-annotated dataset that is used to train the model) and the *target domain* (the test dataset that the model is applied to) and the family of work that addresses this knowledge transfer is known as transfer learning. While general transfer learning methods cover a wide range of cases, *domain adaptation* only refers to those cases where the task remains the same while

the distributions vary between domains. For example, when training a phishing detection model, enabling knowledge transfer between the URL dataset used to train the model and the dataset used to test or evaluate the model falls under domain adaptation, where the task of phishing detection remains the same. Again, the primary objective is to generalise the model learned from the source domain to the target domain while minimising performance loss.

Formally, a domain  $D$  is defined as a triplet consisting of:

- **Feature Space**  $X$ , where each sample belongs to  $\mathbb{R}^d$  and  $d$  is the dimension size of each feature,
- **Label Space**  $Y$ , which can be binary  $\{-1, +1\}$  or multi-class  $\{1, \dots, K\}$ ,
- **Joint Probability Distribution**  $p(x, y)$ , determining the relationships between features and labels.

Given a *source domain*  $S$  and a *target domain*  $T$ , training data is drawn from  $S = \{(x_i, y_i)\}_{i=1}^n$ , where  $x_i \in X$  are the input features and,  $y_i \in Y$  are the labels. Similarly, the target dataset consists of  $T = \{(z_i, u_i)\}_{i=1}^m$ .

### 2.2.2 Fundamental work

The general assumption of classical machine learning is that  $p_S(x, y) = p_T(x, y)$ . However, in domain adaptation, this assumption needs to be relaxed. Instead, domain adaptation seeks to minimise the discrepancy between  $p_S(x, y)$  and  $p_T(x, y)$  through adaptation techniques that align distributions while preserving useful source knowledge. However, the techniques used depend on the type of domain shift observed, which can be categorised as prior shift, covariate shift and concept shift. Below we describe each of the following and relate some fundamental work done in each area.

**Prior shift.** This category refers to the scenario where the class prior probabilities differ while the conditional distributions remain the same. That is,

$$p_S(y) \neq p_T(y), \quad (2.1)$$

$$p_S(X|Y = y) = p_T(X|Y = y). \quad (2.2)$$

An example of prior shift occurs in disease detection [47] when a classifier is trained during a disease outbreak. During the outbreak, the probability of the occurrence of the

disease would naturally be higher than later when the probability of the disease would be lower ( $p_S(y) > p_T(y)$ ), although the symptoms given disease occurrence remain the same.

Japkowicz et al. [48] is one of the early works that described this phenomenon as the class imbalance problem. The paper tested this on multiple classifiers including decision trees, neural networks and support vector machines and suggested simple ways to address the problem. The proposed techniques included resampling to reduce the class imbalance and a cost-modifying method to adjust the relative cost associated with misclassifying each class so as to compensate for the class imbalance ratio. Similarly, Chawla et al. [49] tested the technique of over-sampling underrepresented classes and under-sampling overrepresented classes on Naive Bayes classifiers. Other works such as Zhu et al. [50] and Chan et al. [51] addressed this problem in the area of word sense disambiguation by proposing a bootstrap-oversampling method and an EM-based algorithm respectively.

**Covariate shift.** In this case, the marginal distributions of the features differ while the label-conditioned distributions remain the same:

$$p_S(x) \neq p_T(x), \quad (2.3)$$

$$p_S(Y|X = x) = p_T(Y|X = x). \quad (2.4)$$

Covariate shift occurs for example, when an autonomous vehicle trained only during summer time is made to drive in a winter environment. In the new environment, feature distributions will differ such as snow-covered streets and trees in winter which was not a part of the feature distribution of summer. Regardless of whether a tree is covered in snow or not, however, the decision that the vehicle needs to avoid the object is the same.

Previous work generally addressed this problem by reweighting each training sample based on the ratio of  $\frac{p_T}{p_S}$ . To estimate this ratio, Shimodeira et al. [52] explored a non-parametric kernel density estimation. Zadrozny et al. [53] on the other hand, framed this re-weighting estimation as an estimation of whether a test sample is from the source domain or the target domain during evaluation. This paper considered the covariate shift problem as a sample selection bias problem - where the training sample for a machine learning model does not represent the evaluation sample correctly. Other early work in this category include [54] and [55].

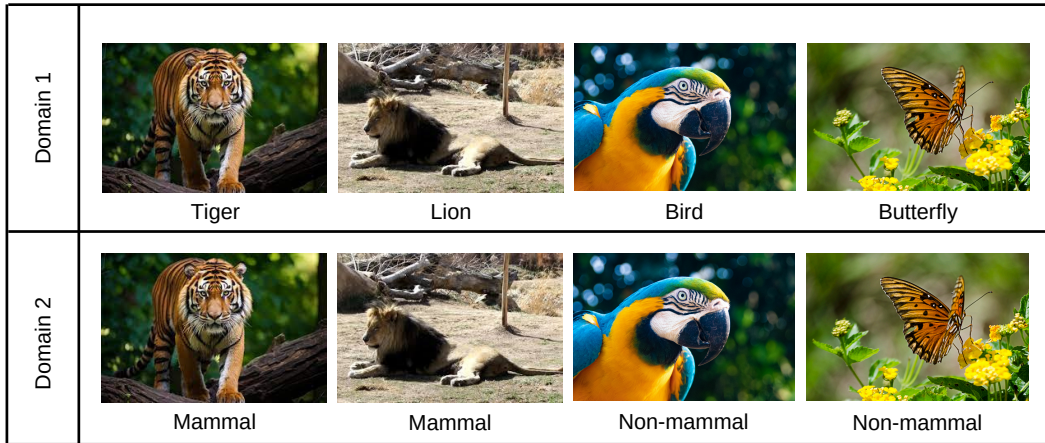


Figure 2.1: Example of concept shift. Here the label distributions vary between the two domains while the feature distributions remain the same.

**Concept shift.** Here, the label distributions change with respect to features, requiring labelled target data for adaptation:

$$p_S(y|x) \neq p_T(y|x). \quad (2.5)$$

Consider a dataset containing animal images, annotated according to species name and another dataset containing the same images but annotated based on whether the animals are mammals or not (Figure 2.1). This scenario represents a concept shift as the conditional distribution of the data labels changed (from species names to mammalian or not) even though the feature distributions remained the same (animal images).

Jiang et al. [24] addressed this by proposing an instance weighting framework. The framework involves multiple instance-based parameters such as a parameter  $\alpha_i$  to estimate the closeness of  $p_T(y_i^S|x_i^S)$  and  $p_S(y_i^S|x_i^S)$  for instance  $i$ . A large value would indicate that  $p_S(y_i^S|x_i^S)$  is a good representation for the instance in the target domain and therefore should weigh higher during training in comparison to an instance with a low  $\alpha_i$ .

### 2.2.3 Application areas

**Computer vision.** A common example from computer vision is when a model trained on images of handwritten digits, such as the MNIST dataset, is used to detect house numbers in images such as the SVHN dataset. Both the source and target data consist of images of digits; however, differences arising from the sourcing of the data introduce a domain shift between the two datasets. Murez et al. [56] for example, proposes a solution building on top of the DenseNet convolutional neural network architecture by adding extra networks and losses which regularises the features extracted from the source domain data. Applied

to SVHN and MNIST, their solution achieves an error rate of 10.4, about 5 points lower than comparable baselines. Other related studies include that of Cicek et al. [57], where a shared embedding is trained to align the joint distribution of the source domain and the target domain. More specifically, the model incorporates an additional joint predictor that models the distribution of both class and domain labels. The encoder is then trained to deceive this predictor, ensuring alignment within same-class samples across domains.

Another related study that addresses cross-domain visual recognition is of Zhang et al [23]. This work proposes the Joint Geometrical and Statistical Alignment (JGSA) framework which aims to reduce both statistical and geometric shifts. To do this, the model learns two coupled projections that project the source and target domain into low-dimensional subspaces where the geometrical shift and distribution shift are reduced.

**Natural language processing.** An example study in the natural language processing space includes Gururangan et al. [25] which investigated the effectiveness of adapting pretrained language models to the target domain task. Specifically, the study proposes a second phase of pretraining in-domain, referred to as domain-adaptive pretraining, where a RoBERTa model is pretrained for a second round on a large unlabelled dataset. The authors evaluated this framework on four different domains, including biomedical and computer science publications, news, and reviews and presented this work as an early demonstration of the effectiveness of pretraining a large language model on the domain of the target dataset before fine-tuning the model for a specific task.

Liu et al. [58], on the other hand, focus on named entity recognition tasks. While cross-domain named entity recognition models exist and can deal with sample scarcity in target domains, the existing NER benchmark datasets do not focus on domain-specific information. To address this, the authors of this paper introduce CrossNER - a cross-domain NER dataset containing NER data from five different domains with specialised entity categories for each domain. Furthermore, the authors provide a domain-related corpus for domain-adaptive pretraining. The study demonstrates that using CrossNER and domain-adaptive pretraining improves NER domain adaptation, outperforming existing cross-domain NER baselines.

**Speech recognition.** Sun et al. [59] on the other hand, applies domain adaptation for robust speech recognition as training and testing mismatch is common in real-world speech recognition tasks. The method proposed by the study learns two joint discriminative classifiers using a deep neural network for two tasks. The main task is to predict the phoneme labels

which is required during both the training and testing time. The second task involves a domain classifier training to discriminate between the source and target domains. This only takes place during the training time. The overall optimisation process involves minimising the label classifier loss and maximising the domain classifier loss simultaneously.

Another similar work by Hsu et al. [60] trains a variational autoencoder on both the source and target domains to learn a latent representation of the speech data. The alignment between the source and target domains is then performed by modifying the latent representations to transform the noisy features in the data that are irrelevant to speech recognition. Evaluated on the CHiME-4 dataset, the method is demonstrated to reduce the absolute word error rate by as much as 35% in comparison to other baselines.

**Activity recognition.** Activity recognition is another field in which domain adaptation plays a significant role. Here the difference in source and target domains often arises when a model is trained on data collected from a specific device or individual and applied to data from a different source. Rokni et al. [61] for example, focuses on domain shifts caused by the addition of new activity sensors in the model training system or varying placement of sensors on different parts of the human body. In such cases, acquiring new labelled data to retrain the model can be difficult. To address this the study proposes an approach where knowledge from existing sensors is used to autonomously learn activity recognition in newly added sensors without the need of any new labelled training data. Evaluating the approach on three experimental datasets, the method achieves an average accuracy of 81.1% representing an improvement of 13.8% due to the automatic labelling of data from newly added sensors.

Khan et al. [62], similarly reduces the need for large volumes of labelled data from the target domain in activity recognition. The authors propose a convolutional neural network-based model named HDCNN which assumes invariant relative distribution of weights in the CNN model layers for the same set of activities. Evaluated on real-world data, HDCNN is demonstrated to achieve over 90% accuracy in scenarios with cross-user differences, and approximately 80% accuracy in the more challenging scenarios such as smartphone-to-smartwatch model transfer, representing improvements of 5 to 20% in comparison to baselines.

## 2.3 Federated learning

As mentioned previously in Chapter 1, this thesis explores federated learning techniques in order to enable intelligence sharing while maintaining data privacy. Organisational threat

intelligence data relevant to phishing detection includes web traffic, access logs and network and endpoint security data which is generally protected through privacy laws and licensing. These would be useful for training robust detection models that otherwise lack a sufficiently broad representation of benign and malicious examples in their training data. The federated learning framework poses a viable solution for training a central model that leverages multiple training data sources while maintaining the privacy of each. In this section, we cover some related work from other areas that apply federated learning for similar purposes and some specific studies that focus on the intersection of domain adaptation and federated learning.

### 2.3.1 Training process

The general federated learning framework consists of  $N$  clients  $\{C_1, C_2, \dots, C_N\}$  where each client possesses a local data  $D_n = \{(x_i^n, y_i^n)\}_{i=1}^{M_n}$  and  $|x^n| = M_n$ . There is a server global model  $\theta_G$  which is broadcast to the clients during each communication round  $t$ . The clients train this model separately using the local data and therefore, each client's local model can be referred to as  $f(x^n, \theta_n)$ . At the end of each communication round, the server collects the clients' model parameters and aggregates them to obtain an updated global model which is broadcast to the clients in the next communication round. The process is elaborated further below.

- Step 1: At the beginning the server initialises the global model  $\theta_G^{t=1}$  which is held at the server.
- Step 2: The latest global model is sent to all participating clients such that the local client model can be represented as  $\{\theta_1^{t-1}, \theta_2^{t-1}, \dots, \theta_N^{t-1}\}$ .
- Step 3: Each client trains the local model and updates the local model parameters such that

$$\theta_n^t \leftarrow \theta_n^{t-1} - \alpha \nabla_{\theta} L_n(f(x^n, \theta_n^{t-1}), y^n)$$

Here,  $\alpha$  is the learning rate and  $L_n(\cdot)$  is the loss for each client.

- Step 4: The server model then collects and aggregates the updated parameters of the clients  $\sum_{n=1}^N \frac{M_n}{M} \theta_n^t$ . This aggregated model is used to update the server model.

$$\theta_G^{t+1} \leftarrow \sum_{n=1}^N \frac{M_n}{M} \theta_n^t$$

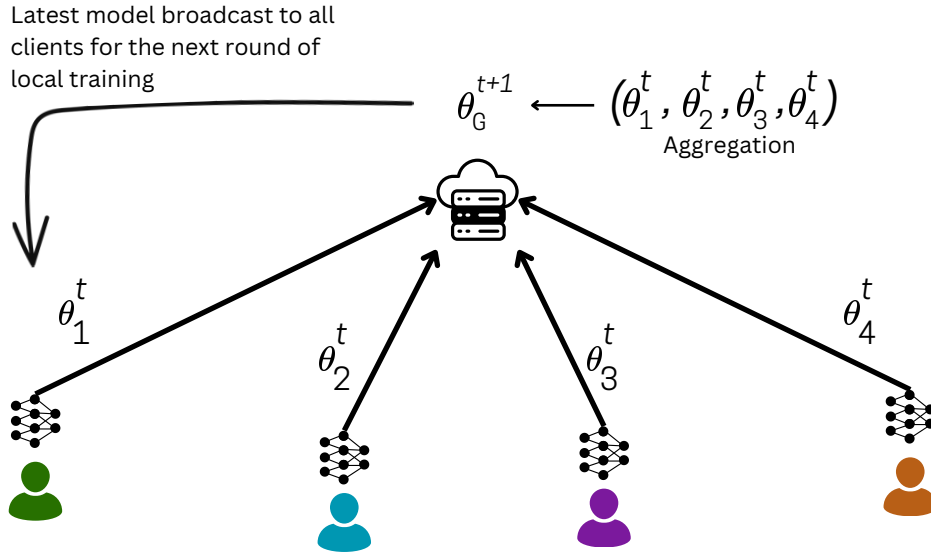


Figure 2.2: General federated learning framework.

- Step 5: Steps 2 to 4 are repeated for  $T$  communication rounds after which the final server global model is distributed to all participating clients.

Figure 2.2 outlines the framework.

### 2.3.2 Application areas

**Medical.** Federated learning is a popular classification method among medical applications as data privacy is crucial in healthcare. Examples include research by Xu et al. [63], where FL was implemented for training a depression detection model while keeping patient data decentralised. Beyond a model training framework that uses data from multiple sources, the study introduces later fusion methods to address time series inconsistencies of multiview data. Compared to other collaborative training frameworks, the study demonstrates that the proposed method predicts depression scores with an accuracy that reaches 85.13%. In comparison to local training, this performance is 15% higher.

In a much earlier work, Kim et al. [64] focus on extracting electronic health record phenotypes (meaningful clinical concepts) using tensor factorisation models. However, this is a challenge when it is necessary to derive joint phenotypes from EHRs of several hospitals as patient-level data sharing is often not possible. The authors address this problem by developing an alternating direction method of multipliers (ADMM)-based method for secure data usage and federated computation. The aim is that hospitals can use this to generate EHR phenotypes using information from multiple hospitals in a secure way. Evaluated on real medical datasets, the method's performance is similar to centralised models trained on combined datasets in terms of accuracy and phenotype discovery. Similarly, Jochems et

al. [65] collected patient data from two hospitals to develop a Bayesian network model for non-small cell lung cancer detection in a distributed setting. Other federated work in medical applications include Kumar et al. [66] who developed a blockchain-based FL model for COVID diagnosis and Liu et al. [67] proposed a NLP-based federated model that trains on clinical notes.

**Recommendation systems.** In the area of recommendation systems, Duan et al. [68] proposed JointRec - a joint cloud video recommendation system that operates at the user level and enables federated training among distributed cloud servers. The study further proposes a method combining low-rank matrix factorisation and 8-bit quantisation to reduce the heavy communication costs in the training process. Caballero et al. [69], on the other hand, focuses on personalised recommendations for student activities and proposes a federated solution based on Eduroam, a Wi-Fi network developed for the academic and research community. The proposed system applies to distributed devices such as smartphones and campus kiosks, and uses user behaviour patterns, class schedules, and activity preferences to provide tailored suggestions for students.

Other federated work in recommendation systems include Wahab et al. [70] which used FL to specifically address the cold-start problem in recommendation systems, where there are no previous ratings available for an item. Here, it is useful to leverage ratings from users from other networks while maintaining their data privacy. Similarly, Tan et al. [71] and Neumann et al. [72] apply FL for content recommendation.

In a more general sense, Muhammad et al. [73] focus on improving federated recommendation systems and propose an alternative for FedAvg, the model aggregation method during each federated global training round. The study proposes FedFast, a sampling method for selecting a diverse set of participating clients for model aggregation, and demonstrates its improved performance in terms of reducing communication costs and training more accurate models earlier on in the federated learning process.

**Intelligent systems.** In a different line of research, Yuan et al. [74] exemplify federated-learning implementation in the development of intelligent transportation systems. This study introduces FedSTN, a federated learning-based spatial-temporal network designed to improve traffic prediction while preserving data privacy. The model leverages graph representation learning to capture spatial dependencies between different traffic nodes and integrates temporal information to improve forecasting accuracy. Other studies exploring federated learning for Internet of Things (IoT) systems and intelligent transport systems (ITS) include Du et al. [75] and Billah et al. [76].

Chen et al. [77] on the other hand, proposes FedHealth, a federated framework for wearable healthcare. The framework applies convolutional neural networks in a federated setting, that is able to train on human activity data from multiple organisations. Similar work utilising healthcare data, collected from IoT devices, to train federated models include Yuan et al. [78] and Liu et al. [79].

**Cyber security.** Within cyber security, intrusion detection research has involved federated learning in multiple cases in order to maintain the privacy of sensitive network information. Tang et al. [80], for example, presented an FL-based method that enables multiple ISPs and other institutions to jointly train while retaining local data. The paper evaluates the method on the CICIDS2017 network intrusion detection dataset and demonstrates performance very close to centralised models. Zhao et al. [81] proposed a similar solution for intrusion detection, however, using a long short-term memory (LSTM)-based model instead.

Liu et al. [82], on the other hand, focused on the security of vehicular networks. The study proposed offloading the intrusion detection model to the edge devices, which in this case involved connected vehicles and roadside units, and further utilised blockchain for secure storage and sharing of the training models. Similar studies in this area include Chelapandi et al. [83] and Basnet et al. [84]. In a similar line, Li et al. [85] introduce DeepFed for intrusion detection in industrial IoT environments. The method employs a hybrid model that combines convolutional neural networks and gated recurrent units to enable multiple cyber-physical systems to train a comprehensive intrusion detection model. Mothukuri et al. [86] and Ghimire et al. [87] exemplify similar work in applying federated learning for intrusion detection in IoT systems.

### 2.3.3 Data heterogeneity

Traditionally, federated learning settings assume that the local data for each client follows the same distribution—that is, that the data is independent and identically distributed (IID). However, in practical scenarios, the data distribution is inconsistent across clients, and various studies have aimed to overcome the performance drops caused by this data heterogeneity.

Li et al. [88], for example, focuses on the effect of low-quality training samples such as data involving skewed label distributions and low content diversity. The paper proposes a framework that enables clients to select the optimal sample combination under a monetary budget while preserving local data privacy. The framework is evaluated on a real AIoT system and demonstrates faster convergence and better performance than existing solutions.

Jeong et al. [89], on the other hand, addresses non-IID client data distribution by proposing a framework where each client trains a generative model. This generative model learns to generate samples to augment the local data such that the resulting dataset becomes more homogenous (towards IID). The study further proposes a federated distillation method to reduce the communication payload during the training process. Through multiple experiments, the solution is shown to reduce communication overhead by a factor of 26 while achieving 95-98% test accuracy compared to normal federated learning.

With the development of transformer models, Qu et al. [90] explored the effectiveness of the self-attention mechanism in overcoming distribution shifts in heterogeneous data. Their work presents the first extensive comparison between convolutional neural networks and transformer models for this purpose, demonstrating the performances of these two types of models over a range of federated architectures, heterogeneous data splits and real-world benchmarks. For instance, one type of experiment presented in the study considers three different types of splits of the CIFAR-10 dataset, each representing a different nature of heterogeneity. Split 1 refers to the data shared among clients such that the label distribution is uniform whereas split 2 consists of client data being skewed towards specific classes. The paper evaluates two convolutional neural network models (ResNet-50 and EfficientNet-B5) and two transformer models (ViT and Swin), and demonstrates that transformer models not only overcome catastrophic forgetting during training, they can also accelerate convergence and reach a more optimal final global model under heterogeneous settings.

Another line of work tackles these heterogeneous settings by proposing personalised federated learning schemes [91, 92]. Wang et al. [91] for example, proposes local fine-tuning of the global model by each client. The study demonstrates this using an RNN language model for the keyboard next-word prediction task and shows that the framework benefits a significant proportion of clients in a heterogeneous setting. Yu et al. [92] on the other hand demonstrated local adaptation through three different techniques: local fine-tuning, multi-task learning and knowledge distillation. Hanzely et al. [93] took a slightly different approach involving learning a mixture of the server model and the local models trained by each client. The study introduces an efficient variant of stochastic gradient descent for this training and shows that the proposed method improves federated communication under heterogeneous settings.

In phishing detection, federated learning presents the opportunity to leverage multiple threat intelligence sources while maintaining organisational privacy. However, similar to the data heterogeneity problem discussed above, the domain shift issue in phishing URLs hampers federated learning performance. The two concepts are essentially the same, as they

both refer to the difference in data distributions held by each client or organisation. That is,

$$P_s(x) \neq P_t(x) \quad (2.6)$$

where  $P_s$  and  $P_t$  represent the source and target domain distributions and the distributions of two clients' local data, respectively. In the later chapters, we explore this domain shift problem further and recommend a federated learning-based solution for phishing detection.

## 2.4 Classification using large language models

The final part of this thesis implements large language models (LLMs) for phishing detection. The main aim in this regard was to leverage large language models' knowledge of benign websites and entities to address the problem of false positives in a more direct manner. Trained on a vast amount of internet data, large language models quickly gained significant attention for their performance in a wide range of tasks including question-answering, machine translation and text classification [94, 95, 96]. The accessibility of pretrained models, their superior performance and the ability to interact with them in natural human language, made LLMs a popular alternative for such tasks.

In LLM research, a family of studies developed that studied LLMs' capabilities to learn from the context provided within the query prompt, known as in-context learning. Compared to supervised learning, this paradigm offered the opportunity to apply LLMs as classifiers without requiring a large annotated dataset for model parameter updates. Instead, in in-context learning, a few demonstrative input and output examples are provided within the query in natural language, for the LLM to learn from before providing a result. Brown et al. [97] formalised the concept of using LLM classifiers under in-context learning as 'Few-shot Learning'. The authors propose the framework of querying LLMs in natural language, along with additional sample answers similar to training samples in supervised learning. The paper evaluates variations of this approach including 'zero-shot' (no sample answers provided), 'one-shot' (only one sample answer provided), 'few-shot' (a few sample answers provided) and 'fine-tuning' (fine-tuning a pretrained LLM for the specific task with many training samples). It was shown that LLMs under zero-shot, one-shot and few-shot can perform similarly to completely fine-tuned models, in some cases nearly matching them in performance.

Liu et al. [98] further investigated how to select in-context examples that improve LLM classification performance. The study proposed a retrieval-based model to select samples that are semantically similar to test samples and demonstrates that this method consistently

produces higher classification accuracy than random selection of samples. Lu et al. [99] on the other hand explored how the order of samples affects the classification performance. They established findings such that the sample order affected even the largest language models and that a good permutation for one model could not be transferred to another.

### 2.4.1 Few-shot classification

In this section, we relate some studies that applied few-shot classification in various areas. Heggseimann et al. [100] for example, demonstrate querying LLMs by adding tabular data to the natural-language query string along with a description of the classification problem. The study finds that under the few-shot setting LLMs can even outperform deep-learning-based tabular classification methods on such tabular classification tasks.

Some more domain-specific work in LLM classification include Li et al. [101] who study LLMs in a cancer application - predicting drug pair synergies in rare tissues. The paper proposes CancerGPT, a few-shot framework that demonstrates performance comparable to larger fine-tuned LLMs. In contrast, Yang et al. [102] applied few-shot LLMs in a cross-modal framework for several audio tasks such as speech emotion classification, audio classification, and text-to-speech classification. Van et al. [103], take a different approach and explore LLMs under few-shot multimodal tasks. The authors apply zero-shot prompting with visual language models such as LLaVa and Flamingo to detect hateful memes and perform meme correction tasks.

A recent significant study by Xu et al. [104] applies LLMs in the field of mental health. This work presents an extensive comparison of multiple LLMs on mental health prediction tasks, based on online textual data. The study not only evaluates a variety of LLMs including Alpaca, Alpaca-LoRA, FLAN-T5, GPT-3.5, and GPT-4, but conducts these evaluations under zero-shot prompting, few-shot prompting, as well as instruction fine-tuning. The results showed that LLMs can perform decently under zero-shot and few-shot settings, however, that performance can be significantly improved with instruction fine-tuning.

### 2.4.2 LLMs in cyber security

In the cyber security field, Aghaei and Al-Shaer [105] proposed SecureBERT - a BERT model fine-tuned for security-based tasks. Specifically, the model is trained to understand the relationship between Common Vulnerabilities and Exposures (CVEs) and tactics, techniques, and procedures (TTPs) within the ATT&CK framework. Furthermore, the study introduced the TTPpredictor tool which extracts threat actions from cyber reports and matches

them to MITRE's attack functionality classes. This was then used as training data for fine-tuning SecureBERT, which was demonstrated to achieve 95-98% F1 scores.

More recently, Zhou et al. [106] explore LLMs' few shot capabilities for recognising code vulnerabilities. The study proposes a prompt design for few-shot vulnerability detection, which includes the task description along with information about dangerous CWE types and vulnerable and non-vulnerable code samples. The framework is evaluated using GPT-3.5 and GPT-4 and the performance is compared to CodeBERT - a BERT model fine-tuned specifically for code vulnerabilities. The evaluation demonstrates that using the proposed prompt design, the performance of GPT-3.5 is comparable to CodeBERT, while GPT-4 outperforms CodeBERT by 34.8% accuracy. Pearce et al. [107], on the other hand, examines LLMs' ability to repair vulnerable code. The paper proposes prompt designs which include information collected from other security tools that detect bugs in code. This information feeds the in-context learning of LLMs to produce repaired code.

To address security flaws in protocol implementation, Meng et al. [108] developed an LLM-based protocol fuzzer. Leveraging LLMs' knowledge about protocol message types of well-known protocols, the authors propose CHATAFL - a protocol fuzzing engine that constructs grammars for different message types within a protocol and thereby mutates the messages in a structured way. The authors evaluate CHATAFL on multiple real-world protocols and draw comparisons with state-of-the-art protocol fuzzers such as AFLNET and NSFUZZ. Compared to these two, CHATAFL demonstrates superior performance in state and code coverage. CHATAFL also detects nine previously unknown vulnerabilities in well-known protocol implementations while state-of-the-art fuzzers only detects three or four vulnerabilities.

# Chapter 3

## Phishing URL Detection using Unsupervised Domain Adaptation

As outlined in the scope of this thesis (cf. Section 1.2), this chapter<sup>1</sup> explores domain shifts in phishing URLs. We investigate the distributional differences in diverse phishing URL datasets and demonstrate their effect on existing phishing URL classifiers. To address this domain shift, we propose an unsupervised domain adaptation framework that improves phishing URL classifier transfer performance.

### 3.1 Introduction

A significant proportion of cyber attacks on the internet comprise phishing attempts to harvest sensitive information such as user credentials. Recent reports estimate that phishing emails are the entry points of 91% of all cyber attacks [109]. As such, identifying phishing URLs accurately and efficiently is essential when it comes to securing corporate networks.

The most prevalent approach to detect phishing attempts includes maintaining blacklists of previously identified phishing URLs and giving a cautionary warning or preventing user access to those URLs, such as Google’s Safe Browsing [110] and Microsoft Smartscreen [111]. Nonetheless, these methods can be inefficient. For example, blacklists can take close to an hour to list unsophisticated phishing websites and close to three hours to list phishing websites employing minor evasion techniques such as redirection via URL shorteners [22]. They can also fail to identify phishing attacks employing sophisticated evasion techniques [22]. Many phishing attempts capitalise on the time gap for blacklists to identify a phishing URL to conduct short-lived phishing campaigns.

---

<sup>1</sup>This chapter includes the work in: **F. Rashid**, B. Doyle, S.C. Han, and S. Seneviratne, “Phishing URL Detection Generalisation using Unsupervised Domain Adaptation”, published in *Computer Networks: The International Journal of Computer and Telecommunications Networking*.

To overcome the limitations of basic blacklist-based phishing URL detection, machine learning-based detectors have been proposed [12, 35], which conventionally rely on the features collected from the phishing web page (e.g., page title, images, HTML information). However, these detectors also can be bypassed by more advanced evasion techniques such as *cloaking* where phishing content on a webpage is replaced with benign content when the page is being accessed by a phishing detection crawler [112]. Other more advanced evasion techniques include redirection via shortened URLs and hosting phishing content on compromised legitimate websites. Overcoming these evasion techniques requires specific attention to the finally visited URL itself. This has paved the way to *URL-based phishing detectors* that solely use the features of the URL to predict whether a given URL is a phishing attempt or not [45, 113]. The advantage of these methods is that they do not require visiting the corresponding web page in the background to make the prediction as the features of the target web page are not used. Such solutions have used machine learning methods [45, 4, 3] as well as more recently deep learning methods [114, 1].

Nonetheless, these state-of-the-art models have always been tested only on one dataset where the training and test sets have the same distribution. As we show in our later analysis, their performance is poor across datasets and, as such, they lack generalisability, creating interoperability issues between various networks as well as being vulnerable to new phishing URLs that are perpetually generated to outmanoeuvre classifiers. To overcome this limitation, we conduct statistical tests across multiple URL datasets and identify the features with distribution shifts causing the models to perform poorly. We identify the issue to be a domain adaptation problem, as the distribution shifts are introduced as a result of collecting URLs from different sources. To mitigate this issue, we propose a feature alignment framework based on Unsupervised Domain Adaptation. More specifically, we make the following contributions.

- We evaluate the performance of several state-of-the-art phishing URL detectors across multiple datasets and show that their performance can drop as much as 10%–32% when the test set is coming from a different dataset than the original training set.
- Through statistical tests, we identify the features that are causing the performance drop and the nature of their distributional shifts. For example, we found that distributional shifts in URL length and the classifiers' focus on the presence of some specific words (e.g., 'www') contribute to performance drops, while counterintuitively, the appearance of new words unseen during training does not affect transfer performance for certain models.
- Based on these findings, we propose an Unsupervised Domain Adaptation (UDA)

approach for URL classifiers when training and test sets are not from the same dataset. To the best of our knowledge, this is the first work to propose domain adaptation in the context of URL classifiers.

- We evaluate the performance of our approach using four existing methods and three datasets and demonstrate an increase in F1 score by 0.06 on average and approximately up to 0.2 in some scenarios.

The rest of the chapter is organised as follows. In Section 3.2 we describe related work in phishing detection and also introduce the field of Unsupervised Domain Adaptation. Section 5.3 describes how our framework aligns with Unsupervised Domain Adaptation. Next, using three different phishing datasets, we conduct our analysis of four existing phishing detection models. In Section 3.4 we introduce the datasets we use and in Section 3.5 we show that state-of-the-art phishing detection models have low transfer accuracies when evaluated using datasets other than the one they were trained on. In Section 3.6 we analyse distributional differences of various features between datasets using statistical tests and identify the features that cause a performance drop in transfer performance. Following this we introduce our feature alignment approach in Section 3.7 and evaluate its performance in Section 5.5. Finally, in Section 5.7, we discuss the implications and limitations of our work and also make recommendations for future work before concluding the chapter.

## 3.2 Related work

### 3.2.1 Approaches to phishing detection

Multiple works proposed machine learning-based phishing URL detection methods. These models extract features from the URLs, landing pages, and hosting information and train machine learning classifiers to predict whether a given URL is phishing or benign. Shahoo et al. [115] broadly categorised these features as blacklist features, lexical features, host-based features, and content-based features. Blacklist features denote the presence of a URL or a slightly modified version of it in a blacklist [44, 116]. Works such as [44, 11, 43, 117] implemented URL lexical features by identifying the presence of words delimited by special characters as well as host-based features such as domain name properties and IP address properties. Several works have also used content-based features such as HTML and Javascript information [12, 13, 35, 36, 18], as well as textual and visual contents [118, 119, 34]. Others further incorporated web-page hyperlinks information [120, 121] such as calculating the ratio of the most common anchor link to the total number of links on the

page. In contrast to these, Tan et al. [14] proposed graph-based methods by leveraging web-page hyperlinks and page-linking data. Finally, several works also used additional features, such as domain age information from WHOIS or page rank from the Alexa top domains list, derived from third-party sources [13, 122, 123, 19, 18].

### 3.2.2 URL-based phishing detection

Most of the above-described features other than the URL features, are difficult to obtain in real-time and such methods can be vulnerable to evasion techniques [22]. Also, generating content features requires visiting the phishing web page in the background. This poses risks of accidental malware download and retaliatory actions, even though it is done in a machine-to-machine manner. Moreover, obtaining hosting information is another difficulty for phishing websites, as they are often short-lived. To address these limitations, URL feature-based methods have been proposed as a complementary alternative. These methods do not rely on features from third-party sources, pose no risk of accidental download of malware, and are not susceptible to webpage-based evasion techniques such as cloaking. Furthermore, these methods can attend to the issue of phishing sites hosted on compromised legitimate websites as they focus only on the URLs. In this chapter, we particularly focus on this specific type of phishing detection method that has been explored in multiple other previous work [3, 1, 2]. Also, it is worth noting that URL-based phishing detection can also be combined with other content-based methods as well. Overall, URL-based phishing detection methods can be broadly categorised into two.

#### 3.2.2.1 Using hand-crafted features

These methods involve extracting hand-crafted features from the URLs such as URL length, word counts, presence of special characters and subdomain counts, and using those features to train machine learning models such as Random Forests and Support Vector Machines [45, 4, 3]. Examples of such methods include CatchPhish [3] and Sahingoz et al. [4], which demonstrated accuracies of 98.25% and 97.98%, respectively. However, both works evaluated their models only on the datasets they collected and introduced in the paper.

#### 3.2.2.2 Using embedding representations

With the advances in deep learning methods, several works proposed deep-learning-based models or deep learning-based representations for phishing URL detection. For instance, Yuan et al. [46] proposed to convert each URL to component-wise character embeddings learned using a SkipGram language model [124] and trained machine learning classifiers

such as XGBoost, Logistic Regression and Random Forest using these embedding representations. On the other hand, Rao et al. [114] fed URL word embeddings into an LSTM architecture, from which a feature layer was extracted to build an ensemble of SVMs.

Other works have implemented end-to-end deep learning pipelines. Both [125] and [1] are similar in that an embedding representation of the URL is used to train a CNN classifier. URLNet [1] first randomly initialises and trains embeddings for each word in an input URL delimited by special characters. Then, it computes embeddings for characters within each word, which are combined with the word-level embeddings by element-wise addition. These embeddings, along with character-level embeddings, are then passed through a CNN architecture for the final prediction. In a similar work, Aljofey et al. [125] also trained a CNN classifier for phishing detection, however, using only character-level embeddings. URLNet reported a true positive rate of 95.58% at a false positive rate of 0.1, while Aljofey et al. [125] reported an F1 score of 95.13%.

More recently, Maneriker et al. [2] proposed URLTran, a model which leveraged state-of-the-art transformer models for phishing detection. Unlike previous approaches, this work uses sub-word tokenised URLs to fine-tune a BERT model [126] that was pre-trained on a massive corpus of natural language documents, and it is considered that this step learns contextual information among URL tokens. The URL tokens are then processed through this model, and the output embeddings are used to train a feed-forward network that generates a phishing prediction. In comparison to URLNet, Maneriker et al. report that URLTran returns a true positive rate (TPR) relative improvement of 21.9% at the false positive rate (FPR) of 0.01%.

*In contrast to these works, we do not propose a new classification model for phishing URL detection. Instead, we investigate how existing models perform across datasets and show that their performances are limited beyond the test sets that are split from the same training dataset. To address this common issue, we propose and evaluate a feature alignment framework based on Unsupervised Domain Adaptation.*

### 3.2.3 Unsupervised Domain Adaptation

Domain adaptation belongs to the larger family of transfer learning, which aims to train a model to perform well on data from a target domain based on knowledge obtained from a different source domain. Within transfer learning, domain adaptation belongs to the sub-category known as transductive transfer learning, in which the domain changes between the source and the target while the task remains the same. As mentioned previously in Chapter 2 (cf. Section 2.2), an example from computer vision is when a model trained on images of handwritten digits, such as the MNIST dataset, is used to detect house numbers in images

such as the SVHN dataset [127]. Both the source and target data consist of images of digits; however, differences arising from the sourcing of the data introduce a domain shift between the two datasets.

Therefore applications of domain adaptation include computer vision [23, 128], natural language processing [25, 129], speech recognition [59, 60], cross-domain recommendation systems [130] and activity recognition [62, 61]. For instance, Khan et al. [62] proposed a domain adaptation solution to address the performance degradation in human activity recognition deep learning models between different devices. Models trained using human activity sensor data (e.g., running and walking) collected using a specific device such as a smart-watch are found to have a performance drop when used to identify human activity from data of the same type but collected from a different device such as a smartphone.

A subcategory of domain adaptation is Unsupervised Domain Adaptation (UDA), which assumes that the source domain is labelled while the target domain is unlabelled. UDA aims to reduce the need for labelled target domain data, as this is often expensive to obtain. Examples of work in this area include [131, 132]. In both studies, a model trained on images captured using a DSLR, for example, is used to recognise the same objects in images captured using a webcam. The different image sources are considered to be different domains, and the target domain (webcam images) is assumed to be unlabelled. Other work, such as [133], used Unsupervised Domain Adaptation for speaker verification systems where a model trained on labelled speech data in the English language is used for speaker verification on unlabelled Chinese language speech data.

*UDA is particularly applicable to the phishing detection problem as phishing URLs are short-lived, and models trained on a particular set of URLs (source domain) need to be effective in detecting newly generated phishing URLs (target domain) in addition to the usual challenges in costs associated with obtaining labelled data. To the best of our knowledge, this is the first work to propose a UDA-based solution to address the cross-dataset portability challenges of phishing URL detection models.*

### 3.3 Our framework

In this section, we introduce how we apply Unsupervised Domain Adaptation (UDA) in phishing URL detection. First, we explain how we adapt the general theoretical UDA framework, and then we provide specific details of our approach.

### 3.3.1 UDA theoretical framework

An outline of the general UDA theoretical framework can be found in [134] where the domain  $D$  is defined as  $D = \{X, Y, p(x, y)\}$ . For our phishing detection problem, each dataset represents a different domain. Therefore  $X \subset \mathbb{R}^d$  is a subset of a  $d$ -dimensional space representing the URL input domain of a unique dataset, and  $X = x_1, x_2, \dots, x_n$  where  $n$  is the number of samples and each  $x_i$  is a  $d$ -dimensional representation of a URL sample. For classifiers that use hand-crafted features [3, 4],  $d$  represents the total number of hand-crafted features - examples of which include the URL length or the word count of a URL. For models such as [1, 2] that use embedding representations,  $d$  is the total number of tokens considered as input from each URL sample; that is, the sequence length.

As phishing detection is a binary problem, the label space  $Y = \{0, 1\}$ , and  $p(x, y)$  represents the joint probability distribution over  $X$  and  $Y$ . As stated in [134], in Unsupervised Domain Adaptation, a function  $f$  is learnt that maps the input to the label space,  $f : X \rightarrow Y$ , by minimising the expected risk:

$$R_T(h) = \mathbb{E}_{(x,y) \sim P_s} \left[ \frac{P_T(x)}{P_S(x)} l(f(x), y) \right], \quad (3.1)$$

where  $P_s$  is the source distribution,  $P_T$  refers to the target distribution,  $l(f(x), y)$  is a loss function denoting the difference between the model prediction and the original label, and  $R_T(h)$  is the sum of all the disagreements in predictions. Here, the ratio of the two marginal distributions is referred to as the importance weight,  $w(x) = \frac{P_T(x)}{P_S(x)}$ , and is used to reweight the source domain data such that it is made similar to the target distribution.

This process of minimising the discrepancy between the two domains is addressed through various methods of UDA [135, 136, 137, 138]. Most of the existing approaches are mainly deep-learning based. For example, [139] proposes a complete deep feed-forward network that combines three processes - the feature extractor, label predictor, and domain classifier - for domain adaptation. However, this approach can only be applied to networks that are trained using backpropagation. Secondly, these approaches consider the entire input feature space  $x \in \mathbb{R}^d$ . That is, they aim to reduce the distance between the latent feature representations of the source and the target domains and, consequently, fail to reveal the specific low-level features in the original data samples that contribute to domain shift. This is often because the application of these works is in computer vision where extracting meaningful low-level features is difficult.

In contrast, we implement UDA in a more generic way such that it can be applied to both machine learning and deep learning models (cf. Section 3.3.2). Furthermore, we investigate individual features for domain adaptation instead of the entire feature space as a whole.

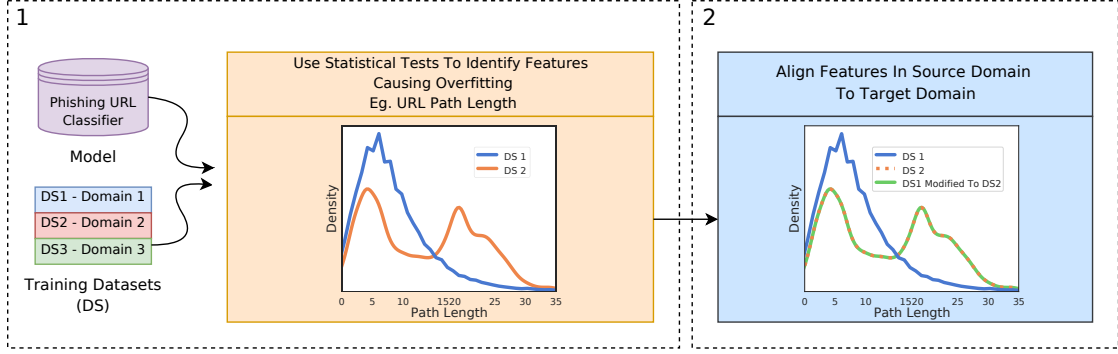


Figure 3.1: Phishing URL Detection using UDA.

By focusing on specific features with different distributions between the source and target domains, we aim to uncover the causes of overfitting that are specific to different datasets as well as the causes that are common across all of them.

Therefore, for our approach, equation 3.1 changes to

$$R_T(h) = \mathbb{E}_{(x,y) \sim P_s} \left[ \frac{P_T(x')}{P_S(x')} l(f(x), y) \right], \quad (3.2)$$

where  $x' \in \mathbb{R}^{d'}$  are the URL samples represented as a subset of the original  $d$ -dimensional feature space, and  $d' \leq d$  is the number of features that we expect to have different distributions between the source and target domains. Therefore, we restrict our reweighting of the source domain ( $w(x)$ ) to only those features that have different distributions between the source and target domains.

### 3.3.2 Phishing URL detection using UDA

In Figure 3.1 we summarise our overall UDA approach for phishing URL classification applicable for both machine learning and deep learning models.

1. For each model we first uncover the  $d'$  exact features that hinder transfer performance between different phishing URL datasets. To do that we use statistical tests to determine if there is a shift between the source and target domains with respect to these features. Further details of this process and the features we identified as overfitting are stated in Section 3.6.
2. Next, we align the features in the source domain to the target domain. We propose different alignment procedures as appropriate for different features and explain this process in further detail in Section 3.7.

## 3.4 Datasets

We use two publicly available URL datasets and one URL dataset compiled by us in this chapter.

### 3.4.1 ISCX-2016 dataset

The ISCX-2016 dataset was collected by Mamun et al. [45]. It consists of benign URLs and various malicious URL types, including malware, spam, defacement, and phishing. Although the dataset contains more than 100,000 URLs altogether, there are only 9,964 phishing and more than 35,300 benign URLs. Thus, to obtain a more balanced dataset, we randomly selected 10,000 benign URLs and combined them with the phishing URLs to build the ISCX dataset for this chapter. The phishing URLs in this dataset were sourced from OpenPhish,<sup>2</sup> and the benign URLs were collected by crawling Alexa top domains.

### 3.4.2 EBBU-2017 dataset

The EBBU-2017 dataset was introduced in [4], and it comprises 36,400 benign URLs and 37,175 phishing URLs. The phishing URLs were collected from PhishTank,<sup>3</sup> while the benign ones were collected using the search engine Yandex's search API.<sup>4</sup> The authors constructed a specific list of words with which they queried the Yandex search engine. The highest-ranked URLs that were returned from this query were collected as benign URLs under the assumption that phishing URLs would not be given a high ranking by search engines.

### 3.4.3 HISPARE-Phishstats dataset

Since ISCX-2016 and EBBU-2017 are somewhat older, we created a new URL dataset consisting of benign and phishing URLs obtained from the HISPARE list [140] and Phishstats,<sup>5</sup> respectively. The HISPARE list contains the landing and internal page URLs for around 2,000 top domains that appear on the Alexa top domains list. We collected our benign URLs from this list by extracting an equal number of internal pages for each domain available. As the number of internal pages for each domain differed, we tried to maximise the number of internal pages that was possible to collect and finally obtained 25 URLs for each of 1,850 unique domains to create a benign list of 46,250 URLs.

---

<sup>2</sup>openphish.com

<sup>3</sup>phishtank.org

<sup>4</sup>yandex.com

<sup>5</sup>phishstats.info

We collected phishing URLs from Phishstats, a website that frequently publishes lists of new phishing URLs. We collected the published lists between the 19th of June and the 26th of August 2022, filtered out all duplicate URLs, and randomly selected 46,250 URLs to ensure our resulting dataset was balanced. The regular feed from Phishstats contains a score denoting the likelihood of a URL being a phishing URL such that: 0-2 means likely, 2-4 means suspicious, 4-6 means phishing and 6-10 means definitely phishing. We selected only the URLs with a score of greater than five to ensure that the URLs in our dataset are indeed phishing. Other works that have used PhishStats as a source for phishing URLs [141, 142, 143] did not involve this additional filtering step.

We randomly split these datasets into training, validation and test sets in a 60:20:20 ratio as summarised in Table 3.1.

Table 3.1: Dataset summary

Dataset	Training		Validation		Test	
	Ben	Phish	Ben	Phish	Ben	Phish
ISCX	5,978	6,001	2,043	1,950	1,979	2,014
EBBU	21,804	22,341	7,227	7,488	7,369	7,346
HP	27,740	27,760	9,180	9,320	9,330	9,170

### 3.5 Transfer performance of baselines

Using four state-of-the-art models for phishing URL classification, we first show that their performances do not translate across datasets.

In our selection we prioritised models that used URL features only (cf. Section 5.2). Among the recent works that focus on URL features only, we selected two that use embedding features and two that use hand-crafted features. From models using embedding features, we selected *URLNet* [1] as this was the most highly cited among the available works. As a second model in this category, we selected *URLTran* [2] as it utilises transformer models that have demonstrated significant success in the area of text classification in general. From among the three models using hand-crafted features that we identified in Section 3.2.2.1, we selected the two most recent papers. These are *CatchPhish* [3], and the model by Sahingoz et al. [4] which we refer to as PhishRF. For URLNet and URLTran we reproduced publicly available source code using the three datasets from Table 3.1. For CatchPhish and PhishRF we trained random forest models using features extracted from the datasets according to the original papers. A summary of the models is provided below.

### 3.5.1 URLNet [1]

*URLNet* is a CNN-based model comprising two CNN branches - one processes a character-level representation of the input URL while the other processes a word-level representation. The input to each CNN branch is an embedding representation of the characters and words, respectively, where two separate embedding matrices are learnt for all the unique characters and words in the training corpus. The words of the URL corpus are obtained by parsing the input URL by special characters, and the output of the CNN architectures from the two branches are concatenated and passed onto a fully connected network for classification.

### 3.5.2 URLTran [2]

*URLTran* model involves first fine-tuning a pre-trained BERT masked language model (MLM). This is done using the URL corpus, where each URL is tokenised using a pre-trained BERT subword tokeniser. The tokens are converted to embeddings, masked according to the original BERT masked language modelling specifications, and passed through the BERT model to predict the correct tokens. A dense layer with the number of classes matching the vocabulary size of the corpus is used for prediction.

After fine-tuning this model, the output of this model is used to train a classification network. Specifically, the CLS token output, which acts as a special pooled token representing the entire input sequence, is passed through a fully connected network that leads to a two-class classification layer.

### 3.5.3 CatchPhish [3]

*CatchPhish* uses manually extracted features from the URL as input features to train a random forest model. One type of such features includes count-based features. These features represent the number of possible phishing indicator tokens, such as the count of digits, dots, and hyphens. The other type of features involves the presence of tokens that are rare in benign URLs but frequent in phishing URLs. These features check the presence of either special characters or words likely to appear in phishing URLs such as the '\$' or '\*' characters. These features are generated for the hostname and the whole URL separately. This model also uses TF-IDF features of the training dataset vocabulary and, finally, checks for the presence of brand names that phishing attackers often target. The full list of features can be found in the original paper [3].

Table 3.2: F1 scores for baseline experiments using the URLNet, URLTran, CatchPhish, and PhishRF models. Each model demonstrates high performance when evaluated on the test data collected from the same source as the training data. In contrast, performance drops for URLs from different sources.

Train Sets	Models	Test Sets		
		ISCX	EBBU	HP
ISCX	URLNet	<b>0.99</b>	0.69	0.68
	URLTran	<b>0.99</b>	0.68	0.67
	CatchPhish	<b>0.99</b>	0.68	0.69
	PhishRF	<b>1</b>	0.68	0.67
EBBU	URLNet	0.88	<b>0.98</b>	0.92
	URLTran	0.83	<b>0.99</b>	0.93
	CatchPhish	0.80	<b>0.94</b>	0.88
	PhishRF	0.87	<b>0.96</b>	0.94
HP	URLNet	0.72	0.79	<b>0.99</b>
	URLTran	0.75	0.86	<b>0.99</b>
	CatchPhish	0.73	0.72	<b>0.98</b>
	PhishRF	0.83	0.86	<b>0.99</b>

### 3.5.4 PhishRF [4]

This approach initially conducts data pre-processing on all URLs before extracting hand-crafted features, which are used to train a random forest model. The aim of the data pre-processing is mainly to parse the URL into words and identify similarities between the words and known brand names, and phishing keywords. Some of the features that are extracted thereafter are similar to those of CatchPhish, such as digit count, average word length, longest word length, and the presence of special characters. However, there are a few unique features, such as the count of words that are similar to brand names and phishing keywords and the count of words that contain a random sequence of characters. The full list of features can be found in the original paper [4].

For each model, we performed hyperparameter-tuning for each validation dataset and used the model with the best parameters to obtain predictions for each of the test sets. We provide a summary of these baseline results in Table 3.2 and use the F1 score as a performance metric as it provides a balanced reflection of the true positive and false positive rates. This is important in phishing detection, where high false positive rates are a common issue. Overall, our baseline results reflect that the F1 score is high when testing on the test splits from the same datasets and lower when testing on test splits from different datasets. We hypothesise that the reason for this involves differences in the distributions of the input features across the three datasets. As the input features to each detection model are different, in the next section we analyse the relevant features for each model separately.

## 3.6 Analysis of transfer performance

We next investigate the features among our three datasets that are likely to be overfitting features and potentially contribute to low transfer accuracy for each model, respectively. Our analysis here is two-fold. First, we check whether an existing distributional shift in the URL vocabulary (i.e., the presence of new words) is a source of the problem. This analysis is applicable for the URLTran and URLNet models that use word-level embedding representations and the CatchPhish model that uses TF-IDF values of the words in the vocabulary. We highlight that PhishRF doesn't use any vocabulary-based features. Therefore, this analysis does not apply to it. Second, for CatchPhish and PhishRF, we focus on the distributional differences of the most important hand-crafted features and check whether they cause the accuracy drop.

### 3.6.1 New words and contradictory words

We define two types of new words that can affect the cross-dataset performance: *new words* and *contradictory words*. Here, we highlight that the training and test sets are coming from two different datasets.

- **New words:** words that appear in the test set samples but not in the training set samples.
- **Contradictory words:** words with differing frequency patterns between training and test sets (e.g., words that are prevalent in benign samples in the training set but the opposite in the test set).

To obtain the contradictory words, we extract the set of common words for every training and test set pair, identify the majority label for each word by counting the number of benign and phishing URLs that they appear in, respectively, and select the label with the most counts. Then, we shortlist contradictory words by filtering the words with contradicting majority labels between the training and test sets.

For each training and test set pair, we compared the number of correctly and incorrectly classified URLs based on whether they contained a certain number of new words or contradictory words. We observed in general that for the URLNet and URLTran models, the URLs containing new words and contradictory words were more often correctly classified than incorrectly classified. Figure 3.2 illustrates one example of this observation. The figure displays the number of URLs from the ISCX test set that are correctly and incorrectly

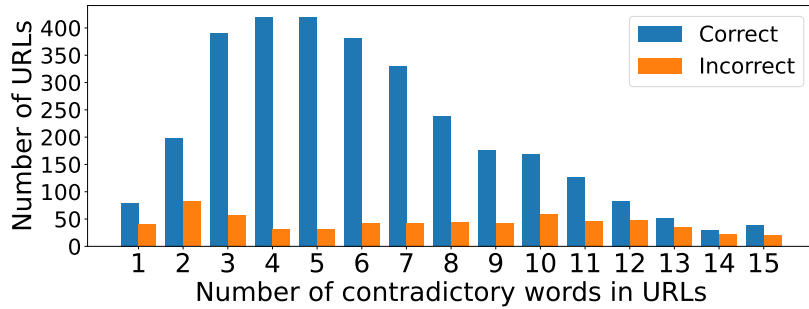


Figure 3.2: Correctly and incorrectly classified ISCX test URLs with contradictory words, by URLTran trained on EBBU.

classified by the URLTran model trained on the EBBU dataset based on the number of contradictory words within the test URLs. We observe that the number of correctly classified URLs is generally greater than the number of incorrectly classified URLs.

To test the statistical significance of this observation, for different counts of new and contradictory words in each URL, we calculated the difference between the number of correctly classified and incorrectly classified URLs and conducted sign tests [144]. The sign test is a statistical test that checks for consistent differences between two samples of observations. That is, the sign test considers the median of the differences between paired observations and checks to statistical significance, whether it is either equal to, greater than or less than an expected value. Thus for our analysis, we apply this test to the values of difference between the number of correctly and incorrectly classified URLs.

As the sign test checks whether the median of a group of values (difference values between the number of correctly and incorrectly classified URLs) is equal to an expected value or not, it is similar to the one sample t-test. However, the t-test assumes a normal distribution for the sample, while our sample of difference values displayed a skewed distribution. Therefore, we selected the sign test, which is non-parametric and does not assume that the data follows any particular distribution.

For each training and test dataset combination, we performed three separate sign tests which tested the null hypothesis that the median difference value is equal to zero against three alternative hypotheses separately:

1. *Two-sided alternative*: the median difference value is not equal to zero
2. *Greater-than alternative*: the median difference value is greater than zero
3. *Less-than alternative*: the median difference value is less than zero

The p-value results of the sign tests are shown in Table 3.3.

Table 3.3: Sign test p-values: Testing whether the median difference between correctly and incorrectly classified URLs containing new words or contradictory words is equal to 0. Significant p-values (less than 0.05) are shown in bold. The results reflect that models like URLNet and URLTran which use URL word embeddings are robust to new words or contradictory words between source and target datasets. In contrast, CatchPhish is observed to have a drop in model performance due to new and contradictory word as it only uses frequency-based information for words tokens.

		New Words				Contradictory Words					
Model	Train	Test	Alternative Hypotheses			Model	Train	Test	Alternative Hypotheses		
			Two-sided	Greater	Less				Two-sided	Greater	Less
URLTran	HP	EBBU	<b>3.91e-03</b>	<b>1.95e-03</b>	1.00e+00	URLTran	HP	EBBU	<b>3.55e-15</b>	<b>1.78e-15</b>	1.00e+00
		ISCX	<b>7.81e-03</b>	<b>3.91e-03</b>	1.00e+00			ISCX	<b>2.83e-06</b>	<b>1.41e-06</b>	1.00e+00
	EBBU	HP	6.25e-02	<b>3.13e-02</b>	1.00e+00		EBBU	HP	<b>6.62e-05</b>	<b>3.31e-05</b>	1.00e+00
		ISCX	2.19e-01	1.09e-01	9.84e-01			ISCX	<b>1.57e-04</b>	<b>7.83e-05</b>	1.00e+00
ISCX	HP	1.25e-01	9.92e-01	6.25e-02	ISCX	HP	<b>5.10e-03</b>	<b>2.55e-03</b>	9.99e-01		
	EBBU	6.54e-02	9.94e-01	<b>3.27e-02</b>		EBBU	<b>6.04e-07</b>	<b>3.02e-07</b>	1.00e+00		
URLNet	HP	EBBU	<b>1.19e-07</b>	<b>5.96e-08</b>	1.00e+00	URLNet	HP	EBBU	<b>3.52e-02</b>	<b>1.76e-02</b>	9.96e-01
		ISCX	<b>1.83e-03</b>	<b>9.16e-04</b>	1.00e+00			ISCX	5.81e-01	8.67e-01	2.91e-01
	EBBU	HP	<b>4.34e-03</b>	<b>2.17e-03</b>	1.00e+00		EBBU	HP	<b>2.25e-02</b>	<b>1.12e-02</b>	9.98e-01
		ISCX	<b>3.13e-02</b>	<b>1.56e-02</b>	1.00e+00			ISCX	6.54e-02	<b>3.27e-02</b>	9.94e-01
ISCX	HP	<b>1.54e-03</b>	<b>7.72e-04</b>	1.00e+00	ISCX	HP	3.02e-01	9.41e-01	1.51e-01		
	EBBU	<b>1.69e-02</b>	<b>8.45e-03</b>	9.98e-01		EBBU	6.54e-02	<b>3.27e-02</b>	9.94e-01		
CatchPhish	HP	EBBU	1.25e-01	6.25e-02	9.92e-01	CatchPhish	HP	EBBU	6.54e-02	<b>3.27e-02</b>	9.94e-01
		ISCX	7.03e-02	<b>3.52e-02</b>	9.96e-01			ISCX	1.00E+00	6.23e-01	6.23e-01
	EBBU	HP	2.89e-01	1.45e-01	9.65e-01		EBBU	HP	<b>3.91e-02</b>	<b>1.95e-02</b>	9.98e-01
		ISCX	<b>3.13e-02</b>	<b>1.56e-02</b>	1.00e+00			ISCX	1.09e-01	5.47e-02	9.89e-01
ISCX	HP	2.89e-01	1.45e-01	9.65e-01	ISCX	HP	1.00E+00	5.00e-01	7.46e-01		
	EBBU	1.18e-01	5.92e-02	9.82e-01		EBBU	3.44e-01	1.72e-01	9.45e-01		

We first discuss the results for URLNet and URLTran. For most tests with the two-sided and greater-than alternative, the p-value is statistically significant (less than 0.05). Conversely for all tests with the less-than alternative save for one, the p-values are insignificant (greater than 0.05). This reflects that although the null hypothesis (that the median difference is equal to 0) cannot be rejected in favour of alternative hypothesis 3, it can be rejected in favour of alternative hypotheses 1 and 2, indicating that the median difference between number of correctly classified URLs and incorrectly classified URLs is significantly greater than 0. In other words, of the URLs containing new or contradictory words, the number of correctly classified URLs are significantly greater than the number of incorrectly classified URLs. Thus we conclude that new or contradictory words are not a significant source of transfer error for URLNet and URLTran.

We conducted the same analysis for the CatchPhish model with a slightly different approach with respect to the *new words* analysis. Unlike URLNet and URLTran, we did not consider the number of new words in the URL but instead the ratio of new words to the total number of words in the URL. This is because CatchPhish uses TF-IDF features using a vocabulary built from the training dataset only, and therefore, *new words* appearing in the test sets are not directly accounted for during prediction. However, we hypothesise that the proportion of new words in the URL may have an indirect effect on the model’s predictability as a higher proportion of new words means less TF-IDF information available overall.

The p-value results are shown in Table 3.3. Unlike the URLNet and URLTran models, we observe that most of the p-values for all tests are insignificant, implying that regardless of the alternative hypothesis, the null hypothesis cannot be rejected. This indicates that for the CatchPhish model, new and contradictory words are a source of transfer error. One reason for this may be that CatchPhish only accounts for the frequency-based information for each word token (TF-IDF). Unlike URLNet and URLTran which uses word embeddings, it does not consider the contextual information related to each word in the input URL. As a result, it has lower generalisability and produces more incorrect predictions when new or contradictory words appear in test URLs.

We conclude from these results that distributional differences between the vocabulary of training and test datasets, in the form of new or contradictory words, do not adversely affect the transfer performance of URLNet and URLTran. It does however hamper the transfer performance of CatchPhish. Therefore for URLNet and URLTran, we do not conduct any vocabulary-based modifications in our experiments as outlined in Section 3.7. Instead, we focus on other relevant features such as path length.

### 3.6.2 Distributional differences in features

Next, we focus on the hand-crafted features used in CatchPhish and PhishRF, and explain how influential features were extracted for this analysis. As CatchPhish and PhishRF are random forest models, we use mean decrease in impurity [145] to first identify the top 15 most important features for each model, trained on each of the three training sets. Afterwards, for each model, we selected the features that were common across all datasets. For PhishRF this resulted in seven common top features and for CatchPhish this resulted in six common top features across the three datasets. Table 3.4 lists these top features for the two models. Next, we show that there are statistically significant distributional differences in some of these features among the three datasets that cause a drop in transfer accuracy.

To determine whether the distributional differences of these features between training and test sets are statistically significant, we use the Mann-Whitney U test [146]. The Mann-Whitney U test checks whether two independently sampled groups come from the same distribution. Although the *two-sample t-test* [147] is a popular alternative for this test, it assumes that both samples follow a normal distribution. For our analysis, the two samples are the feature values from the training and test datasets, and an initial check showed that the features do not follow the normal distribution. The distribution of some features is illustrated in Figure 3.3. Thus we selected the non-parametric Mann-Whitney U test, which does not require the samples to follow any particular distribution.

Since the Mann-Whitney U test is known to produce very low p-values when the sample

Table 3.4: Common important features of PhishRF and CatchPhish models trained on HP, EBBU, and ISCX.

**PhishRF**

- 
1. *alexa\_domain*: Is the domain in Alexa Top one million list
  2. *subdomain\_len*: Length of subdomain
  3. *keyword\_count*: The number of keywords in the URL
  4. *raw\_word\_count*: The number of words obtained after parsing the URL by special characters
  5. *path\_len*: Length of path
  6. *std\_word*: Standard deviation of word lengths in the raw word list
  7. *avg\_word\_len*: The average length of the words in the raw word list

**CatchPhish**

- 
1. *http*: TF-IDF of the term http
  2. *num\_dots\_host*: Number of dots in hostname
  3. *www*: TF-IDF of the term www
  4. *len\_domain*: Length of the domain
  5. *avg\_word\_len\_host*: Average word length in the hostname
  6. *longest\_word\_len\_host*: Longest word length in the hostname
- 

size is very large [148], we randomly sample 100 data points for each feature from each dataset for reliable results. We also leave out the binary variable *alexa\_domain* from our analysis since this test supports continuous variables only.

We show the resulting p-values in Table 3.5, for PhishRF and CatchPhish, respectively, where the insignificant p-values ( $> 0.05$ ) are shown in bold. Insignificant p-values imply that the null hypothesis cannot be rejected - that the pair of train and test set feature values can be considered to have the same distribution. On the other hand, significant p-values indicate that the null hypothesis should be rejected in favour of the alternative hypothesis which is that the train and test feature values come from different distributions. Therefore, when the p-value for the train and test set pair of the same dataset is insignificant, but the p-values for the train and test pairs of different datasets are significant, it indicates that overfitting with respect to the given feature may contribute to low transfer performance.

In Table 3.5, for PhishRF, when the test sets come from the same dataset as the train sets, the p-values are insignificant across all common top features, indicating a similar distribution. In contrast, the p-values are mostly significant ( $< 0.05$ ) when the test set is not from the same dataset as the training set. The exceptions to this pattern are the *std\_word* and

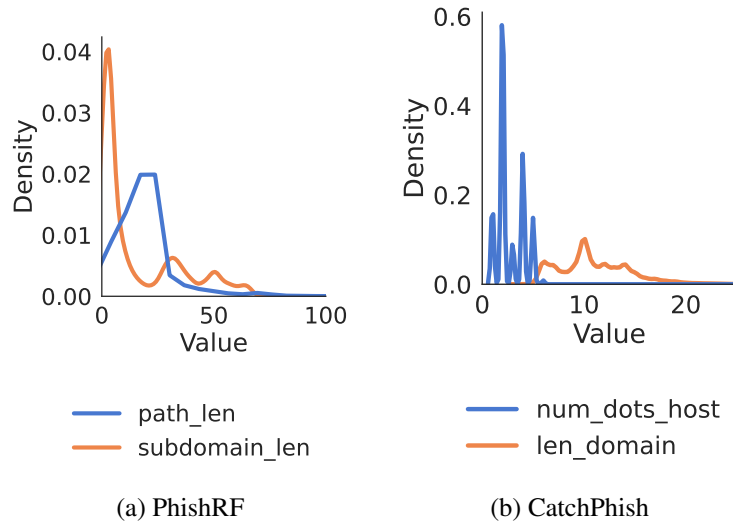


Figure 3.3: Example feature distributions for PhishRF and CatchPhish.

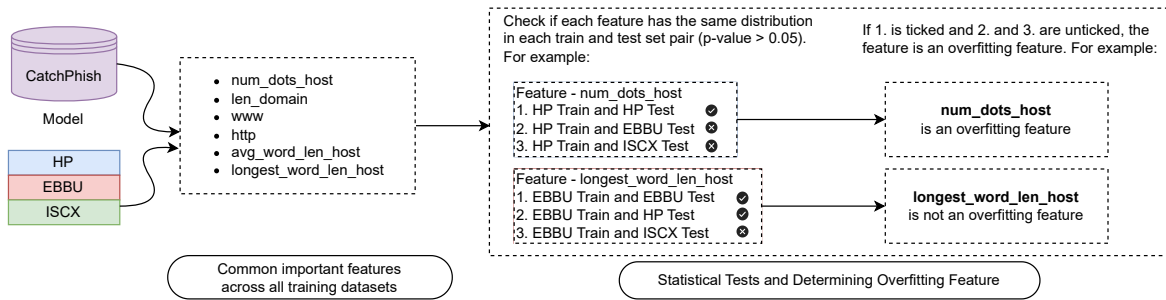


Figure 3.4: Identifying overfitting features for CatchPhish through statistical tests.

*keyword\_count* features. This indicates that if PhishRF overfits with respect to the four other top features, it may cause poor transfer performance on the other datasets. The results for CatchPhish show a similar pattern except for the TF-IDF features for the term *http*.

Overall, for each model, we shortlist a few features which are likely to contribute to low transfer performance if models overfit on them. We shortlist by selecting the features that return insignificant p-values ( $> 0.05$ ) when the test set is from the same dataset as the training set, and significant p-values when the test set is from a different dataset as the training set. For PhishRF these features are *path\_len* and *raw\_word\_count*, and for CatchPhish these features are *num\_dots\_host* and *len\_domain*. In Figure 3.4 we illustrate our method of determining these overfitting features through the statistical tests, using the example of the CatchPhish model.

Table 3.5: Mann-Whitney U test p-values: Testing the null hypothesis that the distributions of the common important features are the same for different training and test set pairs. The insignificant p-values are shown in bold. The results reflect that for most of the top important features the distributions change between the training and the test sets indicating that these features are likely to be causing the poor transfer performance.

		Common top important features - PhishRF					
Train set	Test set	path_len	raw_word_count	subdomain_len	avg_word_len	std_word	keyword_count
HP	HP	<b>5.06e-01</b>	<b>8.93e-01</b>	<b>2.77e-01</b>	<b>8.98e-01</b>	<b>7.44e-01</b>	<b>2.43e-01</b>
	EBBU	5.51e-07	5.02e-05	1.10e-11	2.84e-03	<b>8.95e-01</b>	<b>1.74e-01</b>
	ISCX	3.28e-12	5.74e-18	4.83e-09	2.68e-04	<b>3.39e-01</b>	<b>9.67e-02</b>
EBBU	HP	5.30e-05	6.98e-03	4.56e-15	9.15e-03	<b>1.64e-01</b>	<b>3.77e-01</b>
	EBBU	<b>5.99e-01</b>	<b>5.54e-01</b>	<b>9.77e-01</b>	<b>1.68e-01</b>	<b>2.27e-01</b>	<b>4.31e-01</b>
	ISCX	3.24e-05	3.61e-09	<b>7.30e-01</b>	1.82e-02	<b>1.51e-01</b>	<b>5.36e-01</b>
ISCX	HP	6.93e-14	2.32e-13	2.84e-17	4.91e-03	1.89e-02	<b>2.57e-01</b>
	EBBU	3.09e-12	2.28e-11	1.19e-04	<b>6.96e-01</b>	3.87e-02	<b>6.64e-01</b>
	ISCX	<b>4.64e-01</b>	<b>1.38e-01</b>	<b>7.37e-01</b>	<b>2.88e-01</b>	<b>5.21e-01</b>	<b>8.61e-01</b>

		Common top important features - CatchPhish					
Train set	Test set	num_dots_host	len_domain	www	http	avg_word_len_host	longest_word_len_host
HP	HP	<b>6.17e-01</b>	<b>4.42e-01</b>	<b>3.15e-01</b>	2.57e-02	<b>5.71e-01</b>	<b>1.62e-01</b>
	EBBU	1.04e-06	6.85e-07	<b>1.06e-01</b>	2.38e-10	1.55e-03	6.85e-03
	ISCX	5.63e-11	4.54e-03	5.00e-07	9.52e-10	5.44e-04	5.31e-05
EBBU	HP	1.13e-08	4.11e-11	6.25e-05	<b>5.07e-01</b>	<b>1.34e-01</b>	<b>6.20e-01</b>
	EBBU	<b>9.32e-01</b>	<b>6.26e-01</b>	<b>6.93e-01</b>	<b>2.02e-01</b>	<b>1.22e-01</b>	<b>8.00e-01</b>
	ISCX	3.90e-04	1.66e-04	2.86e-06	6.11e-03	<b>1.02e-01</b>	4.85e-02
ISCX	HP	5.00e-12	2.58e-02	1.06e-06	9.53e-05	1.12e-03	3.51e-04
	EBBU	1.68e-05	7.17e-06	2.93e-02	<b>7.06e-02</b>	3.11e-02	1.11e-02
	ISCX	<b>5.32e-01</b>	<b>6.02e-01</b>	<b>5.23e-01</b>	<b>5.20e-01</b>	<b>9.72e-01</b>	<b>3.47e-01</b>

## 3.7 URL feature alignment

In the previous section, we identified the model-specific features that can hamper cross-dataset performance. Here, we modify the distributions of those features in the source dataset (i.e., training set) to improve the cross-dataset performance. We note that while some of the URL feature-related insights mentioned in this section may be well known, our main contribution in this chapter is our approach to minimising the feature distribution discrepancies between the source and target domains to improve cross-dataset performance.

We outline the features specific to the different models and our alignment process for each of them below. For all models, we removed the ‘http’, ‘https’, and ‘www’ from the beginning of URLs as these are not strong phishing or benign indicators and are likely to serve as noise. We refer to this modification as *removing the URL prefix* in the rest of the chapter.

### 3.7.1 Changing path length

For PhishRF, we found *path\_len* and *raw\_word\_count* to be an important feature with distributional differences among the three datasets. Since these two features are likely to be correlated, we only select the path length. Subsequently, we match the path length distributions of the source training sets to that of the target test set by either truncating or extending

URL paths using a random selection of tokens from the path vocabulary of the training dataset.

---

**Algorithm 1** Modifying the source EBBU benign training path length distribution to match the target ISCX test set path length distribution

---

$V$ : Set of path vocabulary from the EBBU training set  
 $b$ : List of all EBBU training benign URLs in reverse order of path length  
 $S$ : Path length distribution of ISCX target test set  
 $m = |b|$   
 $P_T \leftarrow \text{UPSAMPLE}(S, m)$  ▷ Since  $|S| \leq m$

- 1:  $\text{Modified\_URLs} \leftarrow []$
- 2: **for**  $i$  in  $0, 1, \dots, m$  **do**
- 3:     **if**  $P_T[i] \geq \text{GETPATHLENGTH}(b[i])$  **then**
- 4:          $x = P_T[i] - \text{GETPATHLENGTH}(b[i])$
- 5:          $\text{Modified\_URLs}[i] \leftarrow \text{APPENDPATH}(b[i], x, V)$
- 6:     **else**
- 7:          $y = \text{GETPATHLENGTH}(b[i]) - P_T[i]$
- 8:          $\text{Modified\_URLs}[i] \leftarrow \text{TRUNCATEPATH}(b[i], y)$
- 9:     **end if**
- 10: **end for**
- 11: **return**  $\text{Modified\_URLs}$

---

The overall process we follow to match the path length is described in Algorithm 1 using benign URLs as an example. We first collect the set of words that form the path vocabulary of the source training dataset ( $V$ ). We also separately collect the benign ( $b$ ) and phishing URLs of the training dataset and sort them in decreasing order of path length. Next, we obtain the empirical distribution of target test set URL path lengths ( $S$ ) and compare the frequency of test URLs with the frequency of benign URLs ( $m$ ) from the training dataset. If the target dataset URL frequency ( $|S|$ ) is larger than the benign training URL frequency ( $m$ ), we take a random sample from the target set path length distribution. The size of this random sample matches the number of benign training URLs, and this becomes the desired path length distribution ( $P_T$ ) for the benign training URLs. Otherwise, if the target URL frequency ( $|S|$ ) is smaller than the benign training URL frequency ( $m$ ), we upsample the target path length distribution ( $S$ ) using nearest neighbour interpolation to a size that matches the number of the benign training URLs ( $m$ ) and this becomes the desired path length frequency distribution ( $P_T$ ) for the benign training URLs. In Algorithm 1 we refer to this upsampling approach as the  $\text{UPSAMPLE}(S, m)$  function.

We then iterate through ( $P_T$ ) in reverse order of path length and simultaneously iterate through the set of benign training URLs ( $b$ ). For each URL in ( $b$ ), we calculate the original path length. We refer to this using the  $\text{GETPATHLENGTH}(\text{URL})$  function in Algorithm 1 which counts the number of tokens in the URL path. We then compare the original path

**Examples of extending path length**

<https://www.facebook.com/pacers>  
<https://www.facebook.com/internetnetarchive>  
<https://www.facebook.com/starsportsindia/>  
<https://www.youtube.com/user/espnstar>  
<https://plus.google.com/collection/op4pb>



<https://www.facebook.com/pacersexist>  
<https://www.facebook.com/internetnetarchiveaccountverification>  
<https://www.facebook.com/starsportsindia/filenet>  
<https://www.youtube.com/user/espnstarw7504e07>  
<https://plus.google.com/collection/op4pb01muagz49>

(a)

**Examples of removing hostname dots**

<https://www.goodreads.com/genres/fiction>  
<https://scholar.google.ca/citations?user=S4JEmxgAAAAJ&hl=en>  
<https://www.facebook.com/OfficialPLT/>  
<https://www.udacity.com/school-of-business>  
<https://www.yahoo.com/beauty/>



<https://wwwgoodreadscom/genres/fiction>  
<https://scholargoogleca/citations?user=S4JEmxgAAAAJ&hl=en>  
<https://wwwfacebookcom/OfficialPLT/>  
<https://wwwudacitycom/school-of-business>  
<https://wwwyahoocom/beauty/>

(b)

Figure 3.5: Examples of feature modifications.

length of the URL to the desired path length from ( $P_T$ ). If the desired path length is longer than the original path length we extend the path by appending randomly selected words from the path vocabulary of the training dataset ( $V$ ). This is referred to as the APPENDPATH() function in Algorithm 1. If the desired path length is shorter than the original path length we truncate the path of the URL to the desired path length and this is referred to as the TRUNCATEPATH() function in Algorithm 1. In this way, we match the benign URL path length distribution to the target path length distribution. We conduct the same method for the phishing training URLs and finally obtain a whole training set with a path length distribution that matches the target path length distribution.

Note that our approach does not involve labelled path length information from the target test set. This makes our approach fall under Unsupervised Domain Adaptation. Furthermore, unlike modifying the domain of a URL, modifying the path length in this way does not affect the phishing-relevant information held by the words in a URL. Generally, the domain contains more phishing-relevant information within the hostname and top-level domain, which is why traditional phishing detection models use multiple domain-based word-level features. Examples include checking whether the hostname exists in the Alexa top domains list or not. In contrast, the actual words in the URL path contribute less to the maliciousness of a URL. Therefore adding or removing words from the URL path does not affect the phishing-relevant discriminatory information in the URL.

In Figure 3.5a, we show some example data points from the EBBU training set for which the path length was extended to match that of the ISCX test set path length distribution, and Figure 3.6 illustrates how the path length distributions look once we do the alignment.

Finally, for URLNet and URLTran, the input features are mainly the tokens and the sequence length. As we concluded in Section 3.6.1, new and contradictory words are not the source of error, for these two models, we also focus on modifying the path length as it is a sequence length feature with distributional differences across different datasets. Thus for

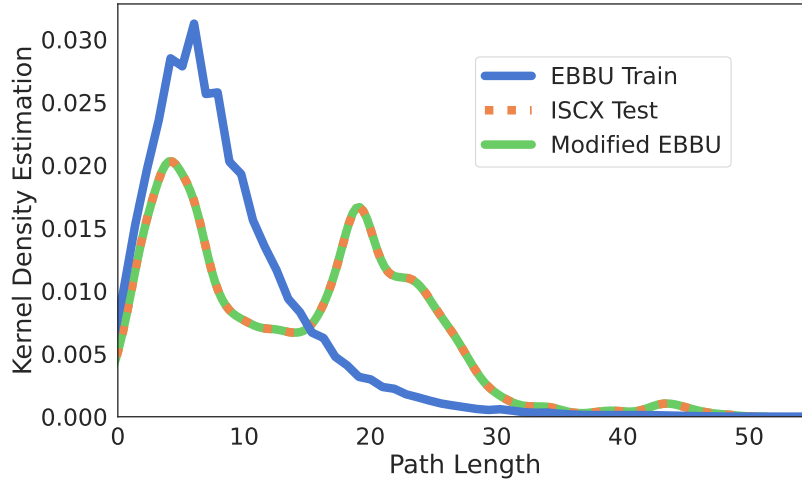


Figure 3.6: Change in EBBU training path length distribution after modifying to match ISCX test set path length distribution.

those two models, we match the distributions in the same way as we do for PhishRF.

### 3.7.2 Removing hostname dots

For CatchPhish, we found the *num\_dots\_host* (number of dots in the hostname) and *domain\_len* (domain length) to be important features with distributional differences among the three datasets. Unlike our approach with path length distribution, we decided against modifying the domain length by altering domain tokens. This is because the domain generally contains strong benign/phishing indicators, such as whether the domain exists in the Alexa top domains list. Therefore, for CatchPhish, we focus mainly on the number of dots in the hostname and modify this by removing all dots from the hostname in the source and target datasets. This is exemplified in Figure 3.5b.

*To summarise, for URLNet, URLTran and PhishRF we **change the path length** and for CatchPhish we **remove the hostname dots**, and for all models we further experiment with **removing the URL prefix**.*

## 3.8 Results

We next discuss the results of applying the feature modifications explained in the previous section. In Table 3.6, we report the F1 scores for the model-specific modifications that returned the best results. That is, we report the combinations of feature modifications that returned the best improvement in F1 scores. As mentioned in Section 3.5, we report the F1 scores as it provides a balanced reflection of the true positive and false positive rates. Overall, out of the 24 scenarios we tested, 19 scenarios resulted in a performance improvement.

Table 3.6: Transfer performance: F1 scores before and after applying feature modifications. Overall our feature alignment method results in an average F1 score improvement of 0.06.

	Train	Test	Before	After	Diff	Modification
URLNet	ISCX	HP	0.68	0.71	0.03	remove prefix+change path
	ISCX	EBBU	0.69	0.74	0.05	remove prefix+change path
	EBBU	HP	0.92	0.88	-0.04	change path
	EBBU	ISCX	0.88	0.88	0.00	change path
	HP	EBBU	0.79	0.86	0.07	change path
	HP	ISCX	0.72	0.85	<b>0.13</b>	remove prefix+change path
<b>Average Increase</b>					0.04	
URLTran	ISCX	HP	0.67	0.78	<b>0.11</b>	remove prefix+change path
	ISCX	EBBU	0.68	0.77	0.09	remove prefix+change path
	EBBU	HP	0.93	0.93	0.00	change path
	EBBU	ISCX	0.83	0.91	0.08	remove prefix+change path
	HP	EBBU	0.86	0.89	0.03	remove prefix+change path
	HP	ISCX	0.75	0.79	0.04	remove prefix+change path
<b>Average Increase</b>					0.06	
PhishRF	ISCX	HP	0.67	0.83	<b>0.16</b>	remove prefix+change path
	ISCX	EBBU	0.68	0.87	<b>0.19</b>	remove prefix+change path
	EBBU	HP	0.94	0.95	0.01	change path
	EBBU	ISCX	0.87	0.87	0.00	change path
	HP	EBBU	0.86	0.91	0.05	change path
	HP	ISCX	0.83	0.84	0.01	remove prefix+change path
<b>Average Increase</b>					0.07	
CatchPhish	ISCX	HP	0.68	0.80	<b>0.12</b>	remove hostname dots
	ISCX	EBBU	0.68	0.75	0.07	remove prefix + hostname dots
	EBBU	HP	0.88	0.90	0.02	remove prefix + hostname dots
	EBBU	ISCX	0.80	0.89	0.09	remove hostname dots
	HP	EBBU	0.72	0.69	-0.03	remove prefix + hostname dots
	HP	ISCX	0.73	0.76	0.03	remove hostname dots
<b>Average Increase</b>					0.05	
<b>Overall Average Increase</b>					0.06	

### 3.8.1 URLNet

For URLNet, the best improvements in F1 score were found to be when training on HP and testing on ISCX and vice versa. When trained on HP and evaluated on ISCX, the F1 score improved from 0.72 to 0.85 which corresponds to a total accuracy improvement from 66% to 84%. When trained on ISCX and tested on HP, the F1 score improved from 0.68 to 0.71 which corresponds to a total accuracy improvement from 52% to 59%. Between the two modification types, the average increase in F1 score for URLNet was 0.04 and the average increase in total accuracy was 6%.

Only when training URLNet on EBBU, there was no improvement in F1 score. That is, after the path length modification, evaluating on HP returned a decrease in F1 score and when evaluating on ISCX, the F1 score remained the same. Further investigation showed

that this was caused by a limitation of the Unsupervised Domain Adaptation approach which is that the approach is agnostic to the class-based feature distribution of the target domain. As seen in Figure 3.7, the original path length distribution of EBBU was such that both its benign and phishing boxplot ranges overlaid the benign and phishing ranges of HP. However, when modifying the path length of EBBU to match that of HP, the Unsupervised Domain Adaptation approach requires that the class-based HP distributions be ignored and instead that the overall EBBU path length distribution be matched to the overall HP path length distribution. As a result, it is possible for some of the original class-specific similarities between the source and target domain to be lost after the feature modification.

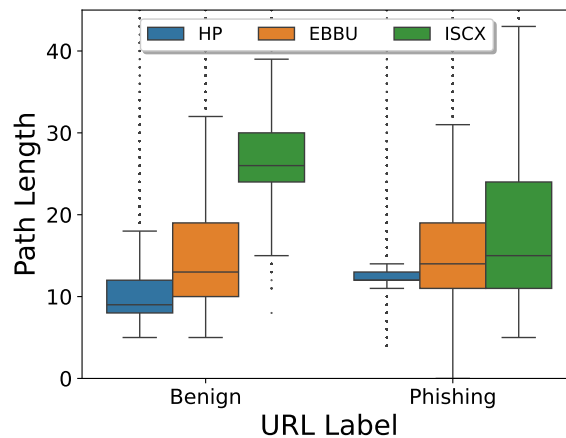


Figure 3.7: Boxplot of path length distributions of the HP, EBBU and ISCX datasets.

In Figure 3.8, we illustrate that this was the case for the EBBU training set when its path length was modified to match that of HP. With respect to matching the EBBU distribution to ISCX, Figure 3.8 shows that after the path length modification, the EBBU benign distribution was still not as similar to the ISCX benign distribution, even though the overall distribution was matched (i.e., Figure 3.6).

### 3.8.2 URLTran

For URLTran, the two greatest improvements in F1 score were found to be when training the model on ISCX. Here the F1 score increased from 0.67 to 0.78 when evaluated on HP, which corresponds to a total accuracy improvement from 52% to 72%. When we evaluated on EBBU, the F1 score improved from 0.68 to 0.77 which corresponds to a total accuracy improvement from 56% to 70%. Overall, the average increase in F1 score for URLTran was 0.06 and the average increase in total accuracy was 7%, indicating the effectiveness of our approach.

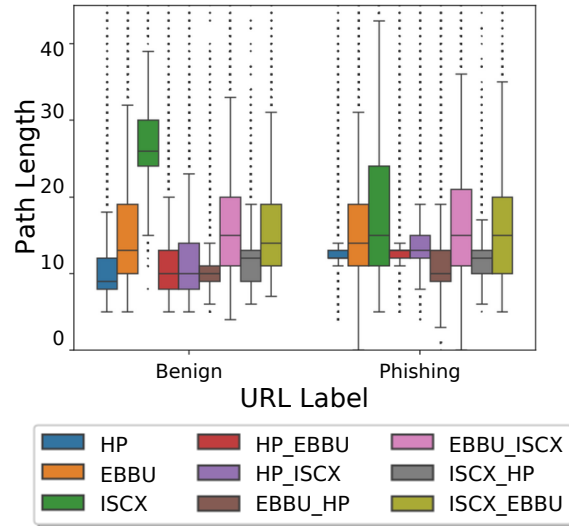


Figure 3.8: Boxplot of path length distributions. For distributions where the path length has been modified, the training set name is followed by the test set name. For example, HP\_EBBU refers to the HP training set modified to match the EBBU test set.

### 3.8.3 PhishRF

For PhishRF, the highest improvement in F1 score was obtained when PhishRF was trained on ISCX with the feature modifications. For this case when we evaluated on HP, the F1 score increased from 0.67 to 0.83 and the total accuracy improved from 51% to 63%. When we evaluated on EBBU, the F1 score improved from 0.68 to 0.87 which corresponds to a total accuracy improvement from 53% to 80%. Overall the modifications resulted in an average F1 score improvement of 0.07 and an average improvement in total accuracy by 7%.

For the case of training PhishRF on EBBU and testing on ISCX, the F1 score remained the same after the feature modifications. One possible reason for this may be that since the classification model is a random forest, the top important features of the model can change due to a distribution change in a feature that was originally important. Therefore, changing the path length of EBBU to match that of ISCX may have caused some other features to become important during training that caused lower transfer performance or at least kept the F1 score the same.

### 3.8.4 CatchPhish

For CatchPhish, the highest improvement in F1 score was also obtained when CatchPhish was trained on ISCX with the feature modifications. For this case when we evaluated on HP, the F1 score increased from 0.68 to 0.80 and the total accuracy improved from 58% to 76%. When we evaluated on EBBU, the F1 score improved from 0.68 to 0.75 and the total accuracy improved from 56% to 67%. Over all cases, the average increase in F1 score was

0.05 and the average increase in total accuracy was 7%.

Only when training on HP and testing on EBBU, the modifications did not improve the F1 score. It is important to note that as the classification model is a random forest model, by removing all the dots from the hostname we essentially remove the feature *hostname* from the model altogether. As a result, there is a rearrangement of the importance assigned to the remaining features during retraining and the resultant important features may adversely affect transfer performance. To investigate this, we observed the top 15 features before applying any feature modification, after removing the hostname dots, and after removing the prefix in addition to removing the hostname dots. We found that before applying any modifications, the top 15 important features included 7 TF-IDF features out of which only two were for words not common in general URLs: ‘uwbrvz8quj’ and ‘page1’. The five remaining words were ‘https’, ‘php’, ‘http’, ‘com’, and ‘www’. After removing the hostname dots, we found that the top 15 important features again included 7 TF-IDF features, however, none of the words were common URL terms. Some examples include ‘limetorrents’ and ‘fundraiser’. After removing the prefix and hostname dots, 11 out of the 15 top features were TF-IDF features and again, none of the words were common URL terms. When the top features depend on terms that are not likely to be found in the test datasets, the transfer performance is naturally likely to decrease.

*Across all models and all training and test set pairs, we conclude that the modifications result in an average F1 score improvement of 0.06 and an average accuracy improvement of 7%.*

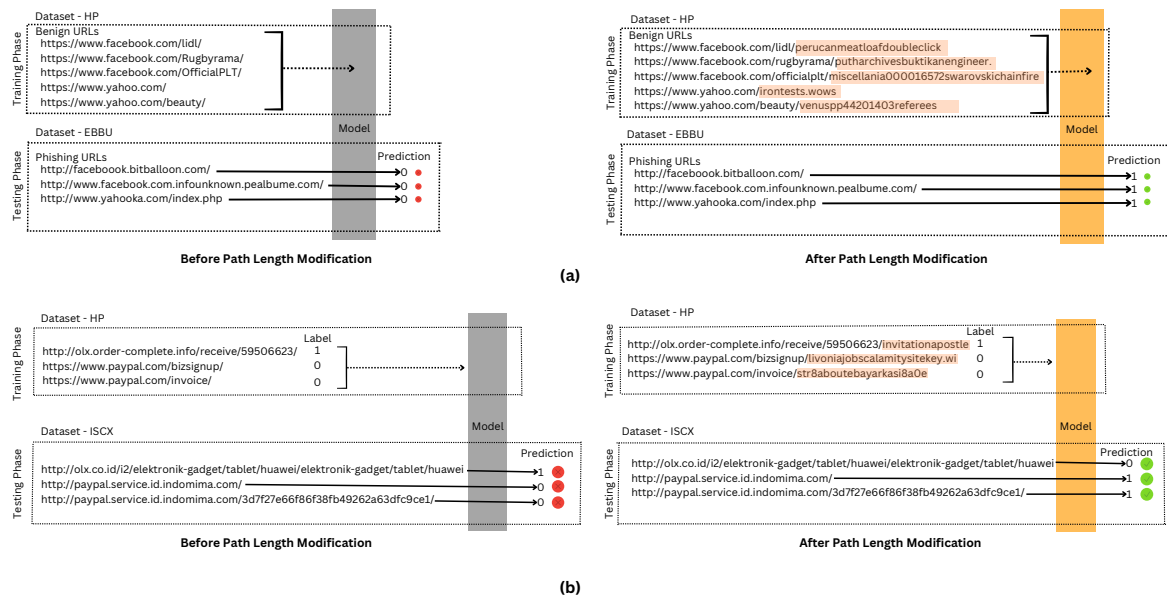


Figure 3.9: Examples of improvement in phishing detection after feature modifications.

### 3.8.5 Examples for performance improvement

Here, we present a few examples of how changing path length returned performance improvement on actual phishing URLs. Originally, the median benign path length in the HP dataset was lower than its median phishing path length (cf. Figure 3.7). Therefore, models trained on HP would learn to classify shorter URLs as benign, and when evaluated on EBBU, some short phishing URLs were misclassified as benign. However, modifying the path length distribution in HP to match that of EBBU removed some of the class-based bias within the HP training URLs and allowed previously misclassified EBBU URLs to be correctly classified. We illustrate this effect in Figure 3.9(a) through some examples. On the top part of the figure, we give examples of some benign URLs in the HP dataset that were extended in path length and on the bottom, we exemplify how this modification resulted in correct predictions at test time.

In Figure 3.9(b) we provide similar examples but this time when testing on the ISCX dataset. We show some examples of URLs with similar domain names but opposite labels in the HP training and the ISCX test set. Before the path length modification the URLTran models incorrectly predicted the ISCX test URLs to have the same label as the URL with a similar domain name in the HP train set. However after the path length modification, the model returned the correct predictions for the ISCX test URLs.

## 3.9 Discussion

Overall, the unsupervised framework we proposed improved the cross-dataset performance of URL classification models. It is important to note that it would not be a fair comparison to compare the cross-dataset performance presented in the previous section with the accuracies presented in existing work involving phishing URL models. This is because while other works mainly propose new models and evaluate them on test sets from the same source domain as the training sets, our main focus was to propose a UDA-based approach to improve the performance of existing models across different target domain test sets.

Due to the fact that our approach is unsupervised, it does not require any effort in manual annotation when building models on target datasets. This can be specifically useful in the area of threat intelligence sharing. As different companies collect URL data in different ways, our framework allows leveraging these datasets in aggregation to build better-performing models. We next discuss the implications of our results, limitations and possible future research directions.

### 3.9.1 Feature differences in URL datasets

Our analysis showed that the low transfer performance of existing URL classifiers is primarily due to variations in path length and URL prefixes across datasets. Although URLs typically follow a standard format, these discrepancies likely arise from different data collection methods.

For example, with regards to the prefix, we observed that benign URLs in the HP dataset mostly began with `https` and phishing URLs mostly began with `http` while most benign URLs in EBBU began with `http`. Models trained on the HP dataset, therefore, would misclassify some benign URLs in EBBU starting with `http` as phishing. One possible reason for such a discrepancy is that some of the benign URLs in the HP dataset were collected by querying the Google Search Engine API with the Alexa top domains, whereas the benign URLs in EBBU were collected using the Yandex Search Engine API. As the Google Search Engine API is known to prioritise HTTPS URLs in search results [149], this may have contributed to the larger proportion of HTTPS benign URLs in the HP dataset in comparison to the EBBU dataset. As discrepancies in the URL prefix are non-indicative of the maliciousness of a URL, they lead to overfitting during training and, therefore, low cross-dataset performance.

Similarly, the path length is another feature affected by how the URLs are sourced. For example, when collecting HP benign URLs, the location for the Google search engine queries was set to the United States. The search results that were collected were only those that returned URLs for web pages in the English language and files such as PDF and Word documents were filtered out. Our analysis showed that this process generally resulted in shorter HP benign URL path lengths than those in the ISCX dataset (cf. Figure 3.7). The ISCX dataset obtained benign URLs using the Heitrix web crawler with the Alexa top domains as the seed input which returned URLs with longer path lengths. A possible reason for this is that, unlike search engines that prioritise returning relevant results efficiently, web crawlers conduct a more comprehensive search through web content and, therefore, are more likely to return longer and more detailed subdirectories in the URL path than search engines. Therefore, a model trained on the ISCX dataset would expect benign URLs to have longer paths and when it is evaluated on the HP dataset, it may misclassify benign URLs with very short path lengths as phishing.

Based on these observations, we highlight that when forming a training dataset for standalone phishing URL classification, it is important to ensure that the path length distribution, in particular, is representative of the general class-based distribution. As the prefixes are generally non-indicative of benignness or maliciousness, those can be removed during pre-processing. Also, we recommend that when releasing URL datasets, it is beneficial to explain in detail the data collection method and any pre-processing methods that have

already been applied.

### 3.9.2 Limitations and future work

Our analysis and approach investigated the transferability of individual features separately. While this revealed specific dataset-related problems that impede transfer performance specifically in phishing detection models, it did not account for inter-correlations among multiple features. The effect of this limitation is observed for CatchPhish in particular where modifying one feature (number of hostname dots) caused the model to overfit on other non-relevant features (cf. Section 3.8.4). This problem could be addressed by integrating correlated-based feature selection methods [150] into the feature alignment framework.

Another limitation relates to the nature of Unsupervised Domain Adaptation which assumes that the target domain is unlabelled. As a result, when using the target domain distribution to modify our source domains, we only matched the overall feature distributions and not the class-specific distributions. We saw in the case of URLNet, that when modifying EBBU path lengths to match that of HP, the original similarity in benign path length distribution between the two datasets was lost and this decreased the transfer performance (cf. Section 3.8.1). Incorporating a limited amount of labelled target data in conjunction with our method may be a possible solution to this issue.

We also observed that between URLNet and URLTran, URLTran responded better to the feature modifications as it produced a higher average increase in F1 score even though the same feature modifications were applied to both. As URLTran is a transformer-based model, it is better able to capture the contextual information of the URL words/tokens during its pretraining stage using masked language modelling. As it is also trained on top of the pre-trained BERT model, which was trained on an enormous amount of text data from the internet, it is also likely to contain contextual information about benign URLs. We believe this is why it can overcome the class-specific differences that arose post feature modifications, which resulted in a decrease in F1 score for one URLNet scenario. As such, it is possible that further performance improvements can be achieved by using Large Language Models (LLMs) such as GPT-3, GPT-4, and LLaMA models [97, 151, 95], that are trained on much larger corpora of internet data and have more model parameters than BERT models.

## 3.10 Conclusion

In this work, we propose a framework for improving the generalisability of phishing URL classifiers based on Unsupervised Domain Adaptation. We demonstrate that state-of-the-art classifiers perform poorly when URLs come from sources different to the classifiers'

original training sets and conduct a statistical analysis to find the specific features with distribution shifts between source and target datasets. Based on these findings, we propose an Unsupervised Domain Adaptation-based framework to align the features across the source and target URL datasets. Our results showed that the URL path length and prefixes are the primary factors causing low transfer performance across the source and target datasets. Our proposed feature-alignment framework resulted in an average F1 score increase of 0.06 and for some scenarios the increase was as high as 0.2. Overall, our work enables facilitating threat intelligence sharing between companies and allows researchers to combine multiple URL datasets obtained from different sources to build better models.

## Chapter 4

# FeUDA-BERT: Federated URL Domain-Aware BERT for Phishing Detection

As outlined in the scope of this thesis (cf. Section 1.2), this chapter<sup>1</sup> explores federated learning for phishing URL detection. We introduce a URL domain-aware self-attention mechanism and a self-attention-based client selection strategy that mitigates client-level data heterogeneity and enhances overall model performance.

### 4.1 Introduction

Our work in the previous chapter elaborates on domain adaptation techniques for phishing URL classifiers where we propose an unsupervised domain adaptation (UDA) framework for phishing URL detection. The approach aligns the feature distributions of the source domain data to that of the target domain and thereby, enables phishing URL classifiers to overcome domain shift challenges when trained and evaluated on URLs collected from diverse sources. In practical scenarios, this approach has two main drawbacks. First, our method is only applicable for one source domain and one target domain whereas several domains need to be leveraged to overcome the domain shift issues and build a suitably robust classifier. Secondly, the UDA approach assumes that the target domain data is accessible.

This presents a difficulty for practical applications because phishing datasets or threat intelligence resources are generally protected by strict privacy regulations. Regulations such as the General Data Protection Regulation (GDPR) [152] in Europe and the Personal Data

---

<sup>1</sup>This chapter includes the work in: **F. Rashid**, B. Doyle, and S. Seneviratne, “FeUDA-BERT: Federated URL Domain-Aware BERT for Phishing Detection”, under review in IEEE Conference on Local Computer Networks (LCN)

Protection Act (PDPA) [153] of Singapore, place strict limitations on how data is collected, stored, and shared by organisations. As a result, most organisations prefer to not share data such as network traffic and previous security incidents specially if they contain any personally identifiable information (PII).

Another limiting factor for threat intelligence sharing is licensing agreements. When contracts are formed between organisations involving the exchange of threat intelligence such as between a security vendor and its client, licensing agreements specify who owns the threat intelligence and to what extent different parties can share or distribute the data. Examples include security providers such as CrowdStrike [154] or Palo Alto Networks [155] who provide threat intelligence to organisations under licenses that restrict the data from being resold or used for other commercial purposes. Similarly, IBM X-Force Exchange provides threat intelligence under licenses that restrict the type of intelligence updates provided to the client and specify the timeframe within which the intelligence can be used internally by the client [156].

Some intelligence sharing groups, such as the US Government's Information Sharing and Analysis Centres (ISACs) [157], operate to enable threat intelligence sharing within specific sectors such as healthcare and financial services. However, they also implement licensing agreements to ensure data is not shared with members outside the sector [158]. On the other hand, security providers working for government or defence organisations are given access to threat intelligence under non-disclosure agreements to ensure the information is not disclosed to third parties.

The above conditions motivate the need for systems where multiple sources of data can be leveraged while maintaining organisational data privacy, and federated learning presents a viable solution. For phishing detection, federated learning enables training a global model using URL datasets from multiple clients while maintaining the local data privacy of each. In this setting, one server global model is broadcast to multiple clients where the model is trained locally on each client's training data. At the end of the local training phase, the participating clients' model updates are sent to the server where the global model is updated based on an aggregation of the clients' model updates. The aggregation follows the Federated Averaging (FedAvg) process that calculates a simple weighted averaging of the clients' model updates, and the updated global model is broadcast to all the clients again for further training rounds with the aim of converging to an optimum global model.

Although naive federated learning successfully avoids sharing local data among clients, it does not address the problem of domain shift among client data. That is, naive federated learning assumes that the clients' local data is independent and identically distributed (IID).

However this assumption is not valid for practical settings. Clients' data features often follow different distributions exhibiting covariate shifts, and the label distributions represented by each client may also exhibit prior shifts (cf. Section 2). This statistical heterogeneity hampers federated model convergence and final model performance [88, 159].

As explored in the previous chapter, different URL datasets exhibit discrepancies in feature distributions which is referred to as covariate shift (cf. Section 2). In a federated setting for URL classification, this breaks the IID assumption for client data as different clients would represent different marginal feature distributions. Previous studies have addressed non-IID federated learning by suggesting data augmentation or sample selection methods aiming to achieve homogenous distributions[88, 90]. Some have suggested curriculum learning-based sample selection and client selection to overcome local data heterogeneity and improve model convergence speed [160]. Others have suggested personalisation methods to align each client more towards its local dataset [93, 161].

However, personalisation methods are not suitable for phishing detection as the aim is for each client to be robust to diverse attacks rather than attacks only aligned to their own local data distribution. Furthermore, existing non-IID studies are more geared towards image datasets rather than text data, let alone URLs. To the best of our knowledge, our work in this chapter presents the first to propose a federated solution for overcoming data heterogeneity specifically for phishing URL detection. We propose FeUDA-BERT, a Federated URL Domain-Aware BERT classifier. Underscored by phishing domain knowledge and findings from the previous chapter regarding URL domain shifts, our model overcomes client data heterogeneity using a URL domain-aware self-attention component and a clustering-based client selection method inspired by curriculum learning.

More specifically, we make the following contributions:

- We propose a privacy-preserving BERT-based phishing URL classifier that overcomes client data heterogeneity specific for phishing URLs. Our proposed model, FeUDA-BERT, is a federated model comprising a URL domain-aware self-attention component based on phishing domain knowledge and URL domain shift insights from the previous chapter. This component serves to give higher weight to the URL domain, which generally contains stronger indications about its maliciousness, and less importance to other URL components such as the URL path length. Furthermore, we introduce a client selection method involving self-attention clustering, inspired by curriculum learning.
- We evaluate our model in a federated setting of 100 clients with local datasets demonstrating feature distribution heterogeneity. In comparison to baselines, we do multiple ablation studies and demonstrate that both our URL domain-aware self-attention

component and self-attention-based client selection method help to improve detection performance. Specifically, our method demonstrates faster model convergence with fewer performance fluctuations and an average improvement of 3% compared to baselines.

- Using the Wilcoxon signed-rank test, we demonstrate that the improvements due to our proposed model are statistically significant. That is, we consider the results of our model compared to baseline models and test the null hypothesis that the difference between the two is zero. We conduct this test against all alternative hypotheses (two-sided, greater, and less than alternatives) and use all results to show that FeUDA-BERT's performance is significantly greater than the baseline models.

## 4.2 Related work

### 4.2.1 Federated learning in cyber security

Several studies have been conducted implementing federated learning as a means of privacy-preserved model training. Application areas include medical applications [63, 64, 66, 67], recommendation systems [68, 70, 72, 73], and intelligent systems (IoT, ITS) [74, 77, 79]. In cyber security, a significant application area for federated learning is intrusion detection. Some studies proposed FL solutions to enable multiple ISPs to jointly train detection models [80, 81]. Another line of work is federated anomaly detection including [162, 86]. Others focused on the security of specific intelligent systems such as intelligent transport systems [83, 84, 82], as well as security of IoT systems in general [85, 87].

A few studies among these focus specifically on cyber threat intelligence (CTI) sharing using federated learning. Trocoso-Pastoriza et al. [163] for example propose a framework for extracting several types of distributed CTI including cyber incidents, vulnerabilities and indicators of compromise, and demonstrate its use in training detection models in a decentralised manner. Ongun et al. [164] on the other hand propose CELEST, a federated model that learns patterns from evolving, globally-coordinated threats over HTTP. The authors evaluated the model on university networks and demonstrated its ability to detect harmful HTTP communication, confirmed to be malicious by VirusTotal. Other work implemented federated learning on blockchain-based CTI platforms for maintaining security and privacy [165, 166, 167], while [168] implemented federated model training on distributed CTI specifically for intrusion detection in IoT systems.

## 4.2.2 Federated learning-based phishing detection

A recent study [169] explored phishing detection in a federated setting utilising a recurrent convolutional neural network (RCNN) and a BERT model as classifiers. Focusing on phishing email detection, this work involved extracting email content features for model training. In contrast, studies such as [170, 171, 172, 173] focused on federated threat detection based on URLs only. Khrantsova et al. [170] for example, train a fully-connected neural network in a federated setting. The study extracts URL features such as the number of letter and URL component lengths from more than 700, 000 malicious and benign URLs and creates client data splits in both IID and non-IID settings, where non-IID splits are created based on the label distribution and dataset sizes. Compared to a centralised model, the paper demonstrates that federated learning improves detection rates up to 27%.

Sakazi et al. [171] on the other hand, implement a combination of federated learning and transfer learning. The proposed solution involves an initial federated training step where distributed Bi-LSTM autoencoders are trained using clients' local unlabelled URL data. This step aims to learn the sequential patterns of local URLs in an unsupervised privacy-preserving manner. The locally trained autoencoders are then aggregated at the server level to obtain a global URL encoder which is then used to train a classification network using open publicly-labelled public datasets. The final model therefore is a semi-supervised federated transfer learning framework.

Similar to [171], Li et al. [172] and Wu et al. [173] also train URL encoders in a federated setting. However, these two studies implement BERT encoders. In [173], a parameter-efficient fine-tuning method LoRA (Low-Rank Adaptation) is used to reduce the computational burden for pretraining a BERT encoder. On the other hand, [172] proposes that only the BERT tokenisers are distributed among clients, while the more computationally expensive BERT encoder training occurs at a central server. Therefore, each client trains their own tokeniser using local URL data and the URL encodings are sent to a server to train separate instances of the BERT encoder model through masked language modelling (MLM). These are then aggregated to obtain a global encoder model. This final encoder is then redistributed to the clients for distributed fine-tuning based on clients' local labelled data.

*Given the superior performance of transformer models, our work in this chapter leverages a federated BERT-based URL classifier. However, unlike similar work in existing literature, we focus specifically on overcoming client data distribution shifts in phishing URL detection. In the next section, we relate existing studies addressing domain shifts in federated learning.*

### 4.2.3 Domain shift in federated learning

As mentioned previously, naive federated learning assumes that client local data follows IID distribution, and when this assumption is not met in practical scenarios, federated models suffer from convergence problems and reduced classification performance. To address this, some studies have proposed to change the client data pool from a non-IID distribution to an IID distribution by either generating new samples to augment the local datasets or by careful sample selection [88, 89]. Other studies explored the effectiveness of different model architectures (such as CNN and transformer models) in overcoming distribution shifts in heterogeneous client data [90, 174](cf. Section 2.3.3).

Huang et al. [159] represent significant work in this area by proposing a contrastive learning-based model to overcome domain shifts in a heterogeneous federated setting. The proposed method computes class-based representations called ‘cluster prototypes’ that capture domain knowledge specific to each client’s data, as well as averaged representations called ‘unbiased prototypes’. Through contrastive learning, sample embeddings are then pulled towards respective cluster prototypes while maintaining alignment with the unbiased prototypes to regulate and stabilise the training process. Upon evaluation, the method performed better than multiple state-of-the-art methods, demonstrating approximately 3% and 7% higher accuracy than Federated Averaging for the benchmark datasets Digits and Office Caltech, respectively. Similar studies include [175, 176].

Another group of studies addresses data heterogeneity through personalised federated learning. Here the aim is to leverage other clients’ datasets through federated learning while also maintaining alignment with the local training data either throughout the federated training process or as an additional local training phase at the end of federated training. As mentioned in Section 2.3.3, studies in this area include [92, 91, 93].

Additionally, Kotelevskii et al. [177] proposed FedPop, a Bayesian personalisation approach where each client learns from fixed common population parameters (representing global patterns shared across all clients) and random effects (client-specific variations deviating from the global population). The study further proposes new stochastic optimisation algorithms for federated settings based on Markov chain Monte Carlo methods. Chen et al. [161] on the other hand, proposed an architecture-based solution involving a shared embedding generator that outputs synthetic embeddings based on global representations. These synthetic embeddings are then transformed into client-specific embeddings by a locally optimised network and used to augment the training space of each client.

Taking a different approach, Vahidian et al. [160] introduced the idea of using curriculum learning to address heterogeneous scenarios in FL. Curriculum learning [178] refers

to the ordering of training samples according to learning difficulty to improve model convergence speed and the quality of the local minima obtained during training. In [160], curriculum learning is applied at the client level by ordering the local data according to a dynamic training loss-based scoring function. The study also applies curriculum learning for client selection during the model aggregation step at the end of each federated communication round by ordering clients according to the mean loss of all the local data points at the client. Through extensive evaluations and theoretical explanations, the authors demonstrate that the order in which clients participate in federation, specially in non-IID settings, has a significant effect on the accuracy performance of the final global model. Related studies in this area include Qi et al. [179] who applied a clustering-based curriculum learning on the local training for each client, and Jiménez-Sánchez et al. [180] who also implemented CL at the client training level but focused specifically on breast cancer classification.

*Unlike existing work in federated learning we propose a lightweight solution to data heterogeneity for phishing URL classification. We do not implement personalised FL as clients in federated phishing detection would aim to be robust to diverse attacks rather than be fine-tuned to the local dataset. Instead, our proposed solution is a combination of two components. The first leverages phishing domain knowledge and our findings from the previous chapter to introduce a new URL domain-aware self-attention loss. The second component involves a self-attention weight-based client selection step inspired by curriculum learning. We describe these in detail in the next section.*

## 4.3 FeUDA-BERT

We propose a Federated URL Domain-Aware BERT-based classification model - a lightweight solution to overcome data heterogeneity specific to phishing URL detection. The proposed model comprises the following two novel components: a URL domain-aware self-attention loss component and a client selection method based on self-attention weight clustering, as described below.

### 4.3.1 URL domain-aware self-attention

We modify a BERT-based classifier by integrating a new self-attention-based loss. This modification is based on the fact that generally, the URL domain holds stronger indications of its maliciousness or benignness, while the URL components hold less important signals. This is reinforced by our findings in the previous chapter, which reflected that signals from other parts of the URL contribute to noise during a BERT-based classifier training (cf. Section 3.9.1). That is, the model tends to overfit to unnecessary features like the URL path

length and initial tokens such as ‘http’ and ‘www’. As a result, in a centralised setting, the model suffers from reduced performance when applied to URLs from different sources with varying feature distributions.

Based on these observations, we introduce a target self-attention distribution to regularise the self-attention weights learned during training. This target distribution represents the importance given to tokens at different positions along a URL, with high values for positions likely to contain domain tokens and decreasing values for all other positions.

Note that the original self-attention mechanism [181], known as scaled dot-product attention, is formulated as

$$Attention(K, Q, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (4.1)$$

where  $K$ ,  $Q$ , and  $V$  are different embedding representations of the input text and  $d_k$  is the scaling factor based on the dimension size of  $K$  and  $Q$ . Here,  $softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)$  represents the attention weight matrix that scales the embedding representation,  $V$ , of the input text.

As the attention weight matrix is derived from the  $K$  and  $Q$  matrices, we formulate our ideal self-attention distribution beginning from the same point. Consider  $K_i$  and  $Q_i$  are one-dimensional vectors,  $K_i, Q_i \in \mathbb{R}^{1 \times U}$  where  $U$  is the maximum token length of URLs being used to train a BERT model. We define  $K_i$  and  $Q_i$  as the following skewed Gaussian distributions:

$$K_i, Q_i = \begin{cases} \exp(-\sigma_1 \cdot (m - u)^2), & \text{if } u \leq m \\ \exp(-\sigma_2 \cdot (u - m)^2), & \text{if } u > m \end{cases} \quad (4.2)$$

where  $u = \{1, 2, \dots, U\}$  represents the token positions of a URL,  $m$  represents the peak position (which we define based on the most likely position of the URL top-level domain) and  $\sigma_1$  and  $\sigma_2$  are the standard deviations of Gaussian distributions on either side of the peak  $m$ . This way the distribution decays (importance decreases) on either side of the top-level domain. We allow for the decay on either side of the peak  $m$  to be asymmetric as the importance of the URL domain tokens should be higher than that of the URL path and accordingly,  $\sigma_1$  may not be equal to  $\sigma_2$ .

Figure 4.1 illustrates this token importance distribution with an example URL and the relative importance of URL tokens based on their position. It displays that importance increases for tokens in the URL domain and decreases after the top-level domain (‘com’). Similarly tokens at the very beginning of the URL such as ‘http’ and ‘www’ are given low importance as these are not strong indicators of the maliciousness of a URL.

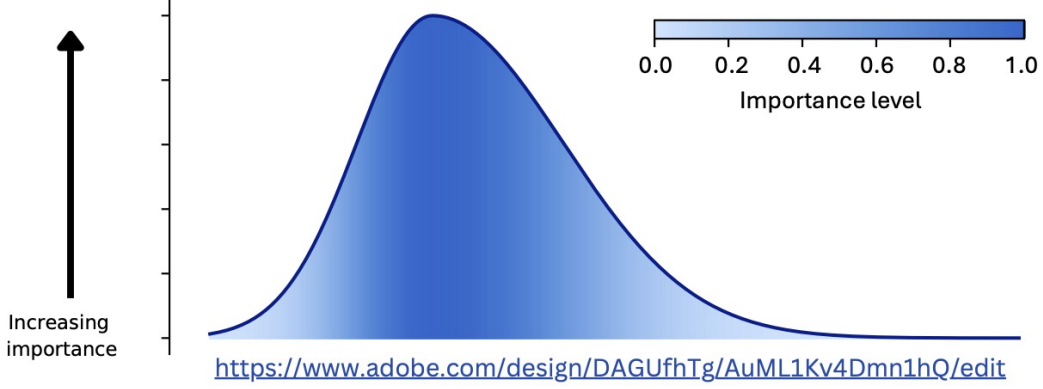


Figure 4.1: Relative importance of URL tokens based on position. We assign higher importance to domain tokens.

Based on the above, we obtain our target self-attention  $A^*$  map (illustrated in Figure 4.2) as a dot product:

$$A^* = \text{softmax}(Q_i K_i^T) \quad (4.3)$$

We incorporate the above into the training of a BERT-based URL classifier by introducing a new loss term based on the difference between  $A^*$  and the learned self-attention matrices. Specifically, for a training dataset  $\mathcal{D} = \{(x_i, y_i)\}$ , we train a URL classifier comprising a BERT encoder with a classifier head. We pretrain the BERT encoder using data samples  $x_i$  and then fine-tune the whole network using labelled samples  $(x_i, y_i)$ .

A typical fine-tuning phase consists of backpropagating the cross-entropy loss,  $\mathcal{L}_h$ , calculated based on true labels  $y$  and the predicted labels  $\hat{y}$ . We further calculate an additional loss based on the difference between our ideal self-attention distribution  $A^*$ , and the self-attention matrix from the final self-attention layers of the BERT model. That is, we calculate the following average:

$$A^{(L)} = \frac{1}{H} \sum_{h=1}^H A_h^{(L)}, \quad (4.4)$$

where  $A_h^{(L)} \in \mathbb{R}^{U \times U}$  is the final layer self-attention matrix from head  $h$ , and  $H$  is the total number of heads. We then compute a Kullback-Liebler divergence loss between  $A^{(L)}$  and  $A^*$  so as to measure the difference between the two distributions

$$\mathcal{L}_{\text{KL}} = D_{\text{KL}}(A^* \parallel A^{(L)}) = \sum_{i,j} A_{i,j}^* \log \frac{A_{i,j}^*}{A_{i,j}^{(L)}}. \quad (4.5)$$

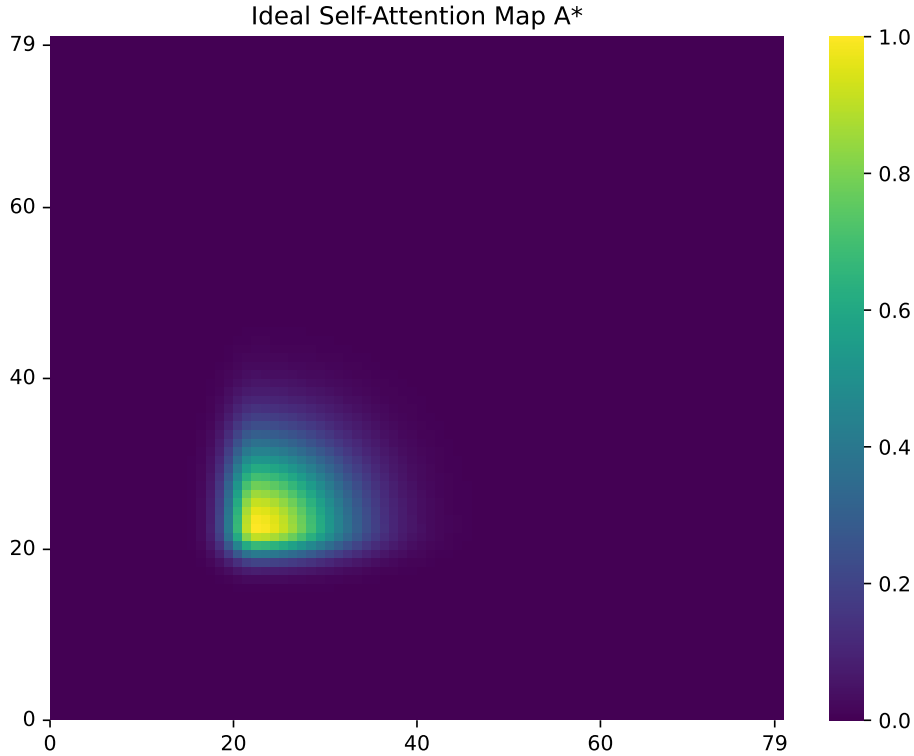


Figure 4.2: Target self-attention map obtained as a dot product of  $K_i$  and  $Q_i$ . The values representing attention related to domain tokens receive higher importance than others.

Based on the above, we define training loss as

$$\mathcal{L} = \mathcal{L}_h + \omega \mathcal{L}_{\text{KL}}, \quad (4.6)$$

where  $\mathcal{L}_h$  is the cross-entropy loss, and  $\mathcal{L}_{\text{KL}}$  is the self-attention matrix loss weighted by  $\omega$ . As this loss is backpropagated through the whole network, the self-attention weights within the BERT model are regularised to follow the distribution of  $A^*$ , ensuring that more importance is assigned to tokens likely to be close to the URL top-level domain while less importance is assigned to tokens further from the top level domain.

### 4.3.2 Self attention-based client selection

We integrate a BERT-based classifier model with the URL domain-aware self-attention loss and implement it in a federated learning setting. In naive FL, the end of each communication round involves aggregating a random selection of clients' local model updates to update the global model. In contrast, we propose a guided client selection approach inspired by curriculum learning. That is, during each communication round we train a subset of clients, randomly selected without replacement, and apply client selection to this subset.

Our proposed client selection method involves clustering clients according to their self-attention weights and selecting clients based on intra-cluster similarity. More specifically, during each communication round we randomly select ten clients without replacement for local training. At the end of each communication round, from each locally updated client model, we concatenate the self-attention matrices from the last layer of each self-attention head and use this to calculate the arccosine pairwise distances between all ten clients.

We perform a hierarchical clustering on these pairwise distances using Ward linkage and cut the hierarchical clustering tree to obtain three clusters. We then sort the clusters in ascending order of cluster size and select a maximum of five clients, beginning from the smallest cluster. We extract three clusters to reflect the three distinct domains in the client data population, which originate from the three separate URL datasets (HP, EBBU and ISCX). This alignment helps ensure each domain is represented and reduces the risk of aggregating client updates representing significantly different domains. Figure 4.3 illustrates this client selection method.

The above client selection method aims to select clients based on similarity in the self-attention matrices learned during training. By performing hierarchical clustering we aim to collect the clients that are most similar to each other, and thereby avoid parameter aggregation between clients with varying feature distributions. By clustering on the self-attention matrices learned using our URL domain-aware component, our client selection method further avoids the adverse effect of noisy data or data heterogeneity in model aggregation. We summarise our FeUDA-BERT model in Algorithm 2.

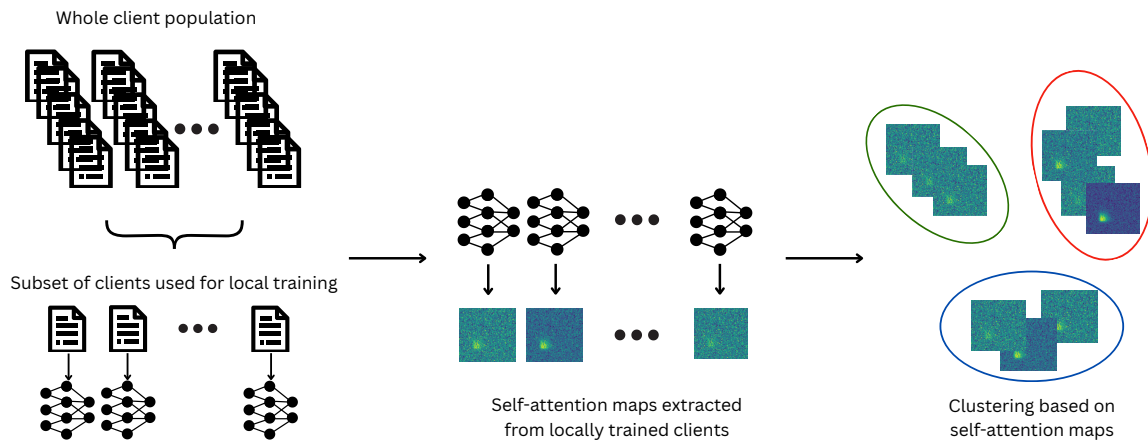


Figure 4.3: Self-attention-based client selection.

---

**Algorithm 2** FeUDA-BERT. The  $K$  clients are indexed by  $k$ ;  $B$  is the local minibatch size,  $E$  is the number of local epochs, and  $\eta$  is the learning rate.

---

**Server executes:**

```

initialize global model parameters  $w_0$ 
for each round  $t = 1, 2, \dots$  do
   $S_t \leftarrow$  (random set of 10 clients)
  for each client  $k \in S_t$  in parallel do
     $w_k^{t+1} \leftarrow$  CLIENTUPDATE( $k, w_t$ )
  end for
   $C_t \leftarrow$  CLIENTSELECTION( $S_t, w_k^{t+1} | k \in S_t$ )
   $m_t \leftarrow \sum_{k \in C_t} n_k$ 
   $w_{t+1} \leftarrow \sum_{k \in C_t} \frac{n_k}{m_t} w_k^{t+1}$ 
end for
CLIENTUPDATE( $k, w$ ): ▷ Run on client k
   $B \leftarrow$  (split  $D_k$  into batches of size  $B$ )
  for each local epoch  $i$  from 1 to  $E$  do
    for batch  $b \in \mathbb{B}$  do
       $\mathcal{L}(w; b) \leftarrow \mathcal{L}_l(w; b) + \omega \mathcal{L}_{\text{KL}}(w; b)$ 
       $w \leftarrow w - \eta \nabla \mathcal{L}(w; b)$ 
    end for
  return  $w$  to server
end for

```

---

## 4.4 Evaluation

### 4.4.1 Federated experiments

Our federated experiments consist of 100 clients, each with local data collected from a single source. That is, using the HP, EBBU and ISCX datasets as before (cf. Section 3.4), and the same data splits as Table 3.1, we employ the training sets such that all clients are allocated equal number of URLs. We ensure that the client’s local data is balanced in terms of labels (benign and phishing) and that each client is allocated data from only one of the three datasets. This way we ensure that the data heterogeneity among clients is only based on differences in feature distribution rather than dataset size or label distribution.

For the domain-aware self-attention component, we define the peak position  $m$  of  $K_i$  and  $Q_i$  according to the most representative top-level domain position of a significantly

Table 4.1: PhiUSIIL Dataset Summary

Training		Validation		Test	
Benign	Phishing	Benign	Phishing	Benign	Phishing
60,514	60,476	20,124	20,207	20,188	20,143

large dataset which is not included in the local client data. For this, we used the PHUSIIL dataset[182]. This dataset consists of over 200, 000 legitimate and phishing URLs collected from Open PageRank Initiative, PhishTank, OpenPhish, and MalwareWorld. Using a 60:20:20 ratio, we formed training, validation and testing splits as described in Table 4.1. We removed any duplicate URLs with HP, EBBU and ISCX, and used a BERT tokeniser to tokenise the training URLs. We then extracted the positions comprising the URL top-level domain tokens and selected the 95th percentile token position as the value for  $m$ . For the values of  $\sigma_1$ ,  $\sigma_2$  and  $\omega$  (weight of the KL-loss), we conducted hyperparameter tuning using the validation set. The final parameter values are 0.1, 0.01, and 0.01 for  $\sigma_1$ ,  $\sigma_2$  and  $\omega$  respectively.

To evaluate the efficacy of our proposed solution, we compare the performance of a BERT classifier with and without the URL domain-aware self-attention component. We also compare our self-attention-based client selection method with random client selection and full network-based client selection. In random client selection we randomly select five clients for parameter aggregation during each communication round. In full network-based client selection we conduct the same hierarchical clustering as self-attention-based client selection, but using the whole set of network weights instead of only the self-attention weights. More specifically, we run the following experiments in federated settings:

- **BERT (random)** - Baseline BERT classifier in a federated setting with random client selection.
- **BERT (SA)** - Baseline BERT classifier in a federated setting with self-attention-based client selection.
- **BERT (FULL)** - Baseline BERT classifier in a federated setting with full network-based client selection.
- **FeUDA-BERT (random)** - BERT classifier with URL domain-aware self-attention component and random client selection.
- **FeUDA-BERT (SA)** - BERT classifier with URL domain-aware self-attention component and self-attention-based client selection.
- **FeUDA-BERT (FULL)** - BERT classifier with URL domain-aware self-attention component and full network-based client selection.

## 4.4.2 Statistical analysis

For all models, we run 20 federated communication rounds with 10 epochs of local training for the participating clients in each round. At the end of each communication round, we evaluated the server global model using the test splits of the HP, EBBU and ISCX datasets. Each experiment was repeated 10 times, and we compared the performances of the models at each communication round, averaged over all three test sets.

Furthermore, to test whether the difference in performance between each model is statistically significant, we apply the Wilcoxon signed-rank test. The Wilcoxon signed-rank test checks whether the probability distribution of two paired samples is equal or not [183]. As we test each model using the same test sets, we consider the averaged model performance results at each communication round to be paired samples. We select this test as it is the non-parametric alternative to the paired t-test, and therefore, does not require strict assumptions such as normality of the sample distributions.

Similar to the sign tests and Mann-Whitney U tests applied in the previous chapter (Section 3.6), for each pair of experimental results, we performed three separate Wilcoxon Signed-Rank tests. Here again, the null hypothesis is that the median difference between the paired samples is zero, while the three alternative hypotheses are the same as those of the sign test in the previous chapter (**cf.** Section 3.6.1).

## 4.5 Results

### 4.5.1 Federated experiments

In Figure 4.4 and Figure 4.5 respectively, we present the F1 scores and accuracy achieved by each model averaged over the three test sets. Since both follow the same pattern we discuss the F1 results below.

**FeUDA-BERT models.** Among the FeUDA-BERT models, we observe that including our clustering-based client selection improves performance compared to random client selection. Although random client selection and full network-based client selection converge faster than self-attention-based client selection, the latter reaches higher F1 scores and maintains this consistently across all communication rounds. After 20 communication rounds self attention-based client selection reaches an average of 0.93 F1 score while random and full network-based client selection reach 0.90 and 0.91 average F1 scores respectively.

**BERT models.** Among the BERT models we observe that self-attention-based client selection initially performs similarly to random client selection before eventually reaching higher

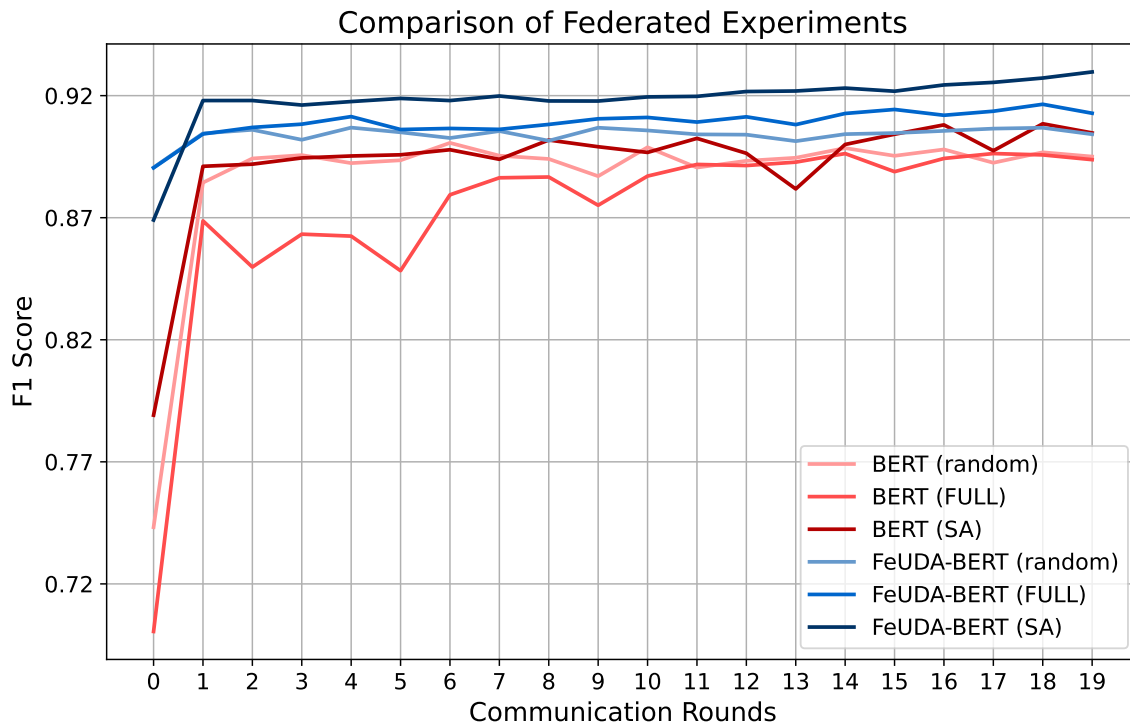


Figure 4.4: F1 score results for federated experiments. Overall the FeUDA-BERT models perform better than BERT models demonstrating the effectiveness of our URL domain-aware self-attention component. Including the self-attention-based client selection mechanism is also found to improve performance.

F1 scores compared to random client selection. Meanwhile, full network-based client selection initially performs worse than the others before eventually matching the performance of random client selection. After 20 communication rounds, self-attention-based client selection obtains an average of 0.91 F1 score while random and full network-based client selection both reach 0.89 average F1 score. These results reflect the superiority of the self-attention-based client selection in comparison to random or full network-based client selection.

**Comparing BERT and FeUDA-BERT models.** In comparison to the BERT models, the FeUDA-BERT models reflect faster convergence to higher F1 scores and significantly fewer fluctuations in performance. Even though BERT with self-attention-based client selection eventually performs similarly to FeUDA-BERT with random client selection, FeUDA-BERT converges to this performance much earlier than BERT and for most of the training phase, consistently performs better.

Overall, we observe that FeUDA-BERT models perform better than the baseline BERT models through all communication rounds. This reflects the effect of the URL domain-aware self-attention component in overcoming data heterogeneity among clients. The results

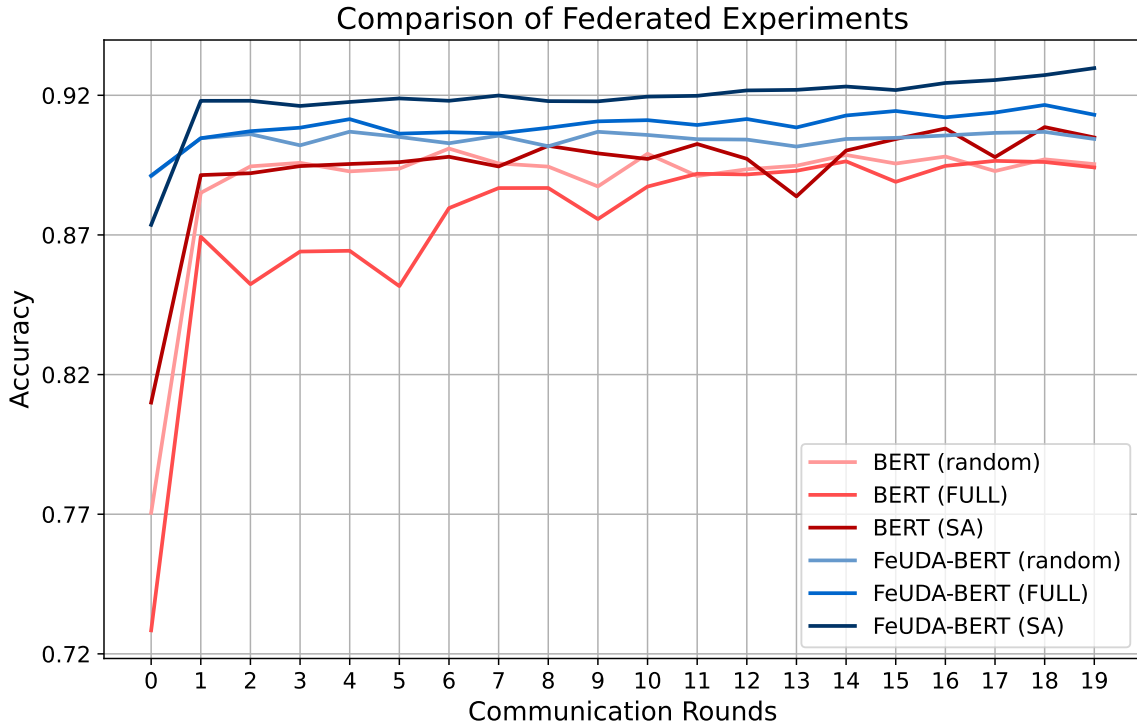


Figure 4.5: Accuracy results for federated experiments.

also reflect that our clustering-based client selection method in combination with the URL domain-aware component results in higher performance in comparison to random client selection. Furthermore, between the two clustering methods for client selection, self-attention-based clustering results in higher performance than full network-based clustering. This is observed in both FeUDA-BERT and BERT independently reflecting that the self-attention maps contain sufficient information to enable effective curriculum learning. However, we find that the superiority of self-attention-based client selection is more consistent in FeUDA-BERT than BERT, reflecting that the combination of this client selection with the URL domain-aware self-attention component provides more reliable superior performance.

## 4.5.2 Statistical analysis

In this section, we present the results of Wilcoxon signed-rank tests assessing the statistical significance of the results from our federated experiments. Table 4.2 displays the p-values of pairwise tests comparing the BERT and the FeUDA-BERT models to assess the significance of the URL domain-aware self-attention component of FeUDA-BERT. Similar to the analysis for sign tests in the previous chapter (see Section 3.6.1), a significant p-value (less than 0.05) means that the null hypothesis can be rejected in favour of the alternative hypothesis. In Table 4.2, we observe that all the p-values for the *two-sided* and the *greater than* alternative are statistically significant, while the p-values for the *less-than* alternative

are insignificant (greater than 0.05). This reflects that for each pairwise model comparison, although the null hypothesis (that the difference between the two models is 0) cannot be rejected in favour of the *less than* alternative hypothesis, it can be rejected in favour of the *two-sided* and the *greater than* alternative hypotheses. In other words, the performance of all the FeUDA-BERT models is significantly greater than the BERT models and the improved performance of the URL domain-aware self-attention component is statistically significant.

Table 4.2: P-values from Wilcoxon signed-rank tests comparing baseline BERT and FeUDA-BERT. The results indicate that the improvement of FeUDA-BERT over BERT in a federated setting is statistically significant.

Sample 1	Sample 2	Alternative Hypotheses		
		Two-sided	Greater	Less
FeUDA-BERT (random)	BERT (random)	<b>1.91e-06</b>	<b>9.54e-07</b>	1.00
FeUDA-BERT (FULL)	BERT (random)	<b>1.91e-06</b>	<b>9.54e-07</b>	1.00
FeUDA-BERT (SA)	BERT (random)	<b>1.91e-06</b>	<b>9.54e-07</b>	1.00
FeUDA-BERT (random)	BERT (FULL)	<b>1.91e-06</b>	<b>9.54e-07</b>	1.00
FeUDA-BERT (FULL)	BERT (FULL)	<b>1.91e-06</b>	<b>9.54e-07</b>	1.00
FeUDA-BERT (SA)	BERT (FULL)	<b>1.91e-06</b>	<b>9.54e-07</b>	1.00
FeUDA-BERT (random)	BERT (SA)	<b>2.61e-04</b>	<b>1.31e-04</b>	0.99
FeUDA-BERT (FULL)	BERT (SA)	<b>1.91e-06</b>	<b>9.54e-07</b>	1.00
FeUDA-BERT (SA)	BERT (SA)	<b>1.91e-06</b>	<b>9.54e-07</b>	1.00

In Table 4.3, we present the results of the pairwise tests that assess the performance significance of the different client selection methods. Among the FeUDA-BERT models, we find that the p values are significant for the *two-sided* and the *greater than* alternatives and insignificant for the *less than* alternatives. These results reflect that the improvement due to our clustering-based client selection method was statistically significant. We further observe that between the self-attention-based client selection and full network-based client selection (FeUDA-BERT (SA) and FeUDA-BERT (FULL)), the higher performance of self-attention-based client selection is also statistically significant.

Among the BERT models in Table 4.3, when comparing BERT (FULL) with BERT (random) we observe that p-values are significant for the *two-sided* and the *less than* alternatives and insignificant for the *less than* alternative. This reflects that the full network-based client selection does not perform significantly better than random selection on the baseline BERT model and this was also represented in Figure 4.4. However, for the other two tests with BERT models, we observe that p-values are significant for the *two-sided* and the *greater than* alternatives and insignificant for the *less than* alternatives. Here again, the results reflect that the self-attention-based client selection provided significantly higher results than random and full network-based client selection.

Table 4.3: P-values from Wilcoxon signed-rank tests comparing client selection methods with baseline BERT and FeUDA-BERT. The results indicate that in general, the self-attention-based client selection demonstrates statistically significant improvement over random or full network-based client selection.

Sample 1	Sample 2	Alternative Hypotheses		
		Two-sided	Greater	Less
BERT (FULL)	BERT (random)	<b>1.05e-04</b>	0.99	<b>5.25e-05</b>
BERT (SA)	BERT (random)	<b>1.21e-02</b>	<b>6.04e-03</b>	0.99
BERT (SA)	BERT (FULL)	<b>4.77e-05</b>	<b>2.38e-05</b>	0.99
FeUDA-BERT (FULL)	FeUDA-BERT (random)	<b>9.54e-06</b>	<b>4.77e-06</b>	0.99
FeUDA-BERT (SA)	FeUDA-BERT (random)	<b>5.86e-04</b>	<b>2.93e-04</b>	0.99
FeUDA-BERT (SA)	FeUDA-BERT (FULL)	<b>7.08e-04</b>	<b>3.54e-04</b>	0.99

## 4.6 Discussion

Next, we discuss the implications and limitations of our approach. Specifically, we mention possible alterations to our URL domain-aware self-attention component. We also discuss the practical scenarios where our method can be applied and possible directions for future work.

**Defining the ideal self-attention map,  $A^*$ .** The results in the previous section demonstrate the effectiveness of our proposed federated phishing URL classification model. We introduced the URL domain-aware self-attention component to regulate the self-attention weights learned during training and therefore regulate the relative importance assigned to URL tokens based on their position. This involved a target self-attention map  $A^*$ , with a fixed peak position  $m$  based on the top-level domain positions of a significantly larger dataset. Another possible way to achieve this is to make  $A^*$  a learnable component. That is, instead of strictly defining the distribution pattern for  $K_i$  and  $Q_i$ , these parameters can be made learnable such that they are learned during training based on the data population represented in the federated setting.

However, without any regularisation on these parameters, the training can become very similar to the baseline BERT model, where the weights of  $K$  and  $Q$  are also learned during training. Our baseline model performance shows that this method is not robust to client data heterogeneity and results in lower overall accuracy. Therefore, it is necessary to regulate the training of  $K_i$  and  $Q_i$  to ensure that the domain tokens are assigned relatively higher importance. Given that a BERT model consists of 12 attention heads, however, learning the  $K_i$  and  $Q_i$  matrices for each head would significantly increase the model complexity. Furthermore, learning the positions of the domain tokens for assigning importance, would depend on the quality of the client data population. If the client datasets are not collectively

well representative of benign and phishing URLs, the final model will not be robust to diverse attacks.

**Scalability.** In our federated setting, we set the number of clients to 100 and maximised the local dataset size for each client while ensuring equal-sized datasets that are balanced in terms of class labels. In doing so, each client received a local dataset of 544 samples from one of three datasets we have used in this thesis (HP, EBBU and ISCX). In terms of the number of clients, this setting is similar to potential settings in existing threat intelligence sharing groups such as the US government’s National Coordinating Center for Communications which comprises around 70 federal government agencies and private sector communications and IT companies [184]. Accordingly, our results reflect the effectiveness of our method for a client population of this size.

Future work can therefore investigate the performance of a higher number of clients with data from more diverse sources. The Financial Services ISAC for example is a global organisation for threat sharing with around 5000 member firms [185]. Besides the large number of clients, a potential federated system for an organisation such as this may face location-based data heterogeneity challenges due to the global scale of the network. Additionally, cross-sector information sharing could introduce distributional shifts due to differences between industries. URL distribution shifts due to specific factors such as this remain an area to be explored in order to improve federated models for threat intelligence sharing.

## 4.7 Conclusion

In this chapter, we propose FeUDA-UBERT, a federated domain-aware BERT-based model for privacy-preserving phishing URL detection. Our contributions comprise a URL domain-aware self-attention component that overcomes data heterogeneity among clients by regulating the relative importance assigned to tokens such that the URL domain receives higher importance. We also introduce a self-attention-based client selection strategy inspired by curriculum learning to achieve higher detection performance. We evaluate our method in a federated setting and demonstrate that each of these two contributions individually improves detection performance in comparison to baselines. Combining the URL domain-aware self-attention component with the self-attention-based client selection further improves detection performance achieving an average F1 score of 0.93. Applying the Wilcoxon signed-rank test, we also demonstrate that the performance improvement due to our proposed solution is statistically significant.

# Chapter 5

## LLMs are One-Shot URL Classifiers and Explainers

As outlined in the scope of this thesis (cf. Section 1.2), this chapter<sup>1</sup> explores embedded knowledge in pre-trained large language models (LLM) for phishing URL detection. Specifically, leveraging LLMs’ reasoning capabilities we propose a one-shot URL classification framework that provides natural language explanations for each prediction. Under extensive evaluations, our one-shot framework provides prediction performances comparable to supervised models and demonstrates alignment with post-hoc explanations of the supervised classifiers.

### 5.1 Introduction

Phishing attacks over emails and other similar channels such as SMS remain a significant concern in cybersecurity and security breaches can be traced back to an initial entry caused by a phishing attack. A recent report sheds light on this issue, noting a 40% increase in phishing attacks between 2022 and 2023, with over 709 million attempts by users to access phishing links blocked by Kaspersky [186]. With attackers’ ability to reach a wide audience in a relatively short time and using short-lived campaigns, defending against phishing remains a highly challenging task that requires multi-faceted solutions.

Existing phishing detection solutions include blacklisting, whitelisting and machine learning and deep learning methods which train on features extracted from the phishing URL, webpage and other external sources [22, 19, 112, 2]. From these, we have focused on URL-only methods in this thesis as they do not require visiting the web page to gather

---

<sup>1</sup>This chapter includes the work in: **F. Rashid**, N. Ranaweera, B. Doyle, and S. Seneviratne, “LLMs are One-Shot URL Classifiers and Explainers”, published in *Computer Networks: The International Journal of Computer and Telecommunications Networking*.

the necessary features [115], and are resilient to evasion techniques such as cloaking [112]. As a result, these methods are more suitable for detecting zero-day phishing campaigns for which prior information is unavailable. Despite the above advantages, these methods suffer from generalisation issues as explored in the previous chapters. URL-only detection methods exhibit high false positive rates, indicating a poor understanding of benign URL indicators [187]. One reason for this issue is the inherent biases in URL datasets, which are collected at different vantage points of various networks.

Moreover, data from one organisation will be biased towards URLs frequently visited by its employees. Sharing data between organisations could potentially address these biases; however, privacy and commercial concerns make it impractical. Although the previous chapter explored privacy-preserving decentralised training, existing URL classification models are also often black boxes, lacking explainability, which is crucial for user awareness and training. That is, given a model's prediction for whether a specific URL is benign or phishing, it is useful to provide users with brief explanations and warnings to help them make informed decisions [188, 189, 190].

Some examples include Outlook [191] and Google [192] which provide users with warnings when they accidentally try to access a suspicious link. Althobaiti et al. [193] further state that in phishing attack detection it is necessary to empower the end-users with more knowledge about a specific situation where a URL is potentially unsafe. Most existing tools that address this are either aimed at experts in the cyber security domain or only provide users with binary decisions about the URL. Such binary advice fails to gain the trust of the users when the decisions have a high false positive rate. For this reason it is important to provide users with simple and easy-to-understand explanations for a certain URL classification.

Recent advances in large language models (LLMs) offer potential solutions to the generalisability and explainability problems of phishing URL classifiers. Trained on vast amounts of data, LLMs excel in natural language text generation and reasoning and are useful for various tasks. Works such as [194, 195, 196] have demonstrated LLMs' predictive performance, which has been enhanced by their recently discovered *in-context learning capabilities* such as few-shot learning [197, 97]. Given that LLMs are trained on a significant portion of internet data, they likely have some knowledge of benign URLs in a broader context. This knowledge can be combined with in-context learning to build URL classifiers, which is the focus of this chapter.

To this end, in this chapter, we propose an LLM-based phishing URL detection framework that integrates both technological and human components of an ideal defence mechanism against phishing campaigns. Our framework leverages LLMs' reasoning and one-shot

learning capabilities for phishing URL classification and provides explanations for each classification. That is, we draw on LLMs’ reasoning capabilities to produce explanations and more accurate URL classifications. As recent work in Chain-of-Thought (CoT) reasoning [198] showed improved accuracy in LLM outputs when prefaced with some reasoning steps, we initially analysed multiple reasoning prompts and their effects on the final URL classification. In our proposed approach, we prompt the LLM to consider the benign and phishing characteristics of a given URL and then make a prediction, including self-reasoning. In our prompt, we include one sample URL, a sample explanation of its benign and malicious characteristics, and a prediction as illustrated in Figure 1. Our results show this approach achieves detection accuracies comparable to supervised deep learning methods trained on large URL datasets. Additionally, the LLM’s reasoning provides natural language explanations that enhance user awareness of benign and phishing URL characteristics, addressing the limitations of existing classifiers that only offer a prediction.

**An Example Prompt and the LLM output**

Prompt: Consider whether the URL seems benign or phishing in 40 words, end sentence with a prediction, the last word being benign/phishing.

**Q:** <http://scholar.google.com.pk/citations?user=IkvxoFIAAAAJ&hl=en>

**A:** The subdomain of this URL is a well-known and reputable internet entity, Google Scholar. Therefore, this URL seems benign.

**Q:** <https://www.rt.com/tags/football/>

Output:

**A:** *The URL directs to a known news website, RT, specifically its section on football. Given its straightforward path and reputable source, this URL appears unthreatening.*

**Prediction:** *benign.*

Figure 5.1: Example URL classification prompt and the output

More specifically, we make the following contributions.

- We propose an LLM-based framework leveraging LLMs’ reasoning and in-context one-shot learning capabilities for phishing URL classification and demonstrate that LLMs are explainable one-shot phishing URL classifiers.
- We evaluate our framework over five state-of-the-art LLMs and three different phishing URL datasets and compare the performance of our framework with existing supervised URL classifiers.

- We demonstrate that our one-shot approach achieves a prediction performance within 0.05 to 0.12 F1 score of a supervised setting using standard URL classifiers. Among the five LLMs evaluated, GPT-4 Turbo performed the best, with an average F1 score of 0.92 in the one-shot setting. This is only 0.07 points below the fully supervised setting, which achieves a 0.99 F1 score.
- We demonstrate the explainability of our classification framework in terms of correctness and overall quality of language. That is, we assess the correctness of LLM self-explanations by comparing the benign and malicious indicators identified through our framework with post-hoc explanations obtained in a supervised setting. We use the G-Eval framework [199] to assess the self-explanations in terms of readability, coherence, and informativeness.
- Finally, we present an extended analysis of the consistency of LLM predictions and the performance of our framework under zero-shot and few-shot settings. Our results show that while GPT-4 Turbo achieves an average F1 score of 0.92 in both the zero-shot and five-shot settings, the average F1 score across all LLMs is lower under the zero and five-shot settings in comparison to one-shot.

The remainder of the chapter is structured as follows: Section 5.2 reviews related work, and Section 5.3 details our overall framework. Section 5.4 outlines our experiment design, while Section 5.5 presents our main results. An extended analysis of these results is provided in Section 5.6. We discuss the implications and limitations of our work in Section 5.7, and conclude the chapter in Section 5.8.

## 5.2 Related Work

### 5.2.1 Phishing URL detection

Automated phishing URL classification is a necessity to safeguard users from accessing phishing sites. Multiple existing solutions utilise machine learning and deep learning methods, which are trained using features extracted from the URLs, landing pages, or external sources. These features, broadly categorised as blacklist features, lexical features, host-based features, and content-based features [115], have been used to train various types of machine learning and deep learning classification models [117, 120, 121]. WHOIS information such as domain age and page rank from the Alexa top domains list are also popular features used in various works such as [13, 122, 123, 19, 18].

One significant disadvantage of using the above features for detecting phishing URLs is the difficulty in obtaining them in real time. Blacklist features can be redundant for short-lived or zero-day phishing URLs, as these exploit the delay in blacklist updates. Acquiring hosting information during the training phase is often impossible for short-lived URLs, and using web-page content features risks accidental malware downloads and is vulnerable to evasion techniques [22]. Consequently, some URL classifiers use only URL-based features, which eliminates reliance on third-party sources, mitigates malware risks and reduces vulnerability to evasion. Our work focuses on these URL-based phishing detectors that use either hand-crafted URL features or an embedding representation of the URL as the primary input feature. Works such as [45, 4, 3] collected hand-crafted features including URL length, word counts and presence of special characters to train Random Forests and Support Vector Machines. In contrast, the URLNet [1] trains a deep learning-based classifier using word and character embeddings to represent the URL, and Maneriker et al. [2] proposed URLTran, a model which leveraged state-of-the-art transformer models for phishing detection. In comparison to URLNet, Maneriker et al. report that URLTran returns a true positive rate (TPR) relative improvement of 21.9% at the false positive rate (FPR) of 0.01%.

The high prediction accuracies reported in existing works were obtained by evaluating URL classifiers on test URLs collected from the same source as the training URLs. When evaluated on data collected from different sources or representing domain shift, these URL classifiers demonstrated low generalisability. Our previous work [187] found that the cross-dataset performance of four URL classifiers (URLNet [1], URLTran [2], CatchPhish [3], and PhishRF [4]) falls by 10%–30% in comparison to their performance on test URLs collected from the same source as the training URLs.

To this end, in this chapter, we propose our one-shot framework for using LLMs as URL classifiers and demonstrate that its performance is better than the cross-data performance of URL classifiers trained in supervised settings. To the best of our understanding, this work is the first to design an LLM-based one-shot URL classification framework. A summary of the works discussed in this section is presented in Table 5.1.

## 5.2.2 Few-shot classification using LLMs

Large Language Models (LLMs) have demonstrated superior performance in a variety of natural language processing tasks, including text generation, question-answering, and text classification [94, 95, 96]. Recently, their generalisability has seen further improvements within the in-context learning paradigm where the LLM performs tasks unseen during training when prompted with task-specific instructions and a few or no (i.e., few-shot or zero-shot) labelled examples [97, 197].

Table 5.1: Phishing URL detection related work summary

Reference	Model Training	URL-only	URL Embeddings	Hand-crafted Features	Webpage Features	External Features
Xiang et al. [13]	Supervised	✗	✗	✓	✓	✓
Geng et al. [122]	Supervised	✗	✗	✗	✓	✓
Rao et al. [123]	Supervised	✗	✗	✓	✓	✓
Alani et al. [19]	Supervised	✓	✗	✓	✗	✗
Chin et al. [18]	Supervised	✗	✗	✗	✗	✓
Mamun et al. [45]	Supervised	✓	✗	✓	✗	✗
Sahingoz et al. [4]	Supervised	✓	✗	✓	✗	✗
Rao et al. [3]	Supervised	✓	✗	✓	✗	✗
Le et al. [1]	Supervised	✓	✓	✗	✗	✗
Maneriker et al. [2]	Supervised	✓	✓	✗	✗	✗
Our work	One-shot	✓	✗	✗	✗	✗

Various works applied the in-context learning capabilities to solve a range of problems, beyond typical natural language processing tasks. For instance, Hegselman et al. [100] and Jaitly et al. [200] applied LLM-based few-shot classification to tabular data. The authors evaluate various serialisation methods and demonstrate the superior performance of few-shot LLMs in comparison to deep learning-based classification on tabular datasets. Other works such as [101, 102, 103] applied few-shot learning for domain-specific classification tasks. For instance, Li et al. [101] evaluated the application of LLMs in biological inference. They prompted LLMs to predict the synergy of drug pairs in rare tissues by including a few labelled examples in the prompt. In contrast, Yang et al. [102] applied few-shot LLMs in a cross-modal framework for several audio tasks such as speech emotion classification, audio classification, and text-to-speech classification, while Van et al. [103] investigated the few-shot capabilities of vision language models on hateful meme detection.

Several works utilised the capabilities of LLMs in cybersecurity. Nonetheless, the efforts have been much more focused on fine-tuning than few or zero-shot settings. For example, Aghaei and Al-Shaer [105] proposed SecureBERT, which is a security-specific fine-tuned BERT model that achieves F1 scores ranging from 95% to 98% in classifying Common Vulnerabilities and Exposures (CVEs) to their corresponding MITRE ATT&CK techniques. The authors also demonstrated that SecureBERT outperforms ChatGPT in Tactics, Techniques, and Procedures (TTP) prediction. Balasubramanian et al. [201] proposed finetuned GPT-3 models as anomaly detection classifiers that achieve more than 99% accuracy.

In contrast to these works, we propose a one-shot LLM classification approach for phishing URL detection, which can also be applied in zero-shot and few-shot settings. Our approach does not require fine-tuning of the language model parameters and demonstrates prediction performances comparable to supervised approaches.

### 5.2.3 Explainability of LLMs as classifiers

The recent success of LLMs in varied domains has additionally motivated researchers to assess LLMs' capabilities as model explainers. Traditionally, explainability algorithms explain black-box model predictions to elucidate the inner model mechanisms. Metrics used to evaluate these algorithms, therefore, assess *faithfulness* (i.e., how well the explanations reflect the true inner mechanism of the model) and *plausibility* (i.e., how well the explanations correlate with human reasoning), aiming to enhance interpretability and trust in the otherwise black-box automated decision-making process. Studies that assess the explainability of LLMs include [202, 203], which implements LLMs as post-hoc explainers of predictions generated by other models. Meanwhile, LLM chain-of-thought reasoning has been considered as self-explanations for their predictions/classifications [204] and in comparison to traditional explanation methods, LLM natural language explanations have been noted for their high plausibility. That is, these explanations are consistent with human logic and reasoning [205, 206]. Nonetheless, due to the complexity of large language model architectures and the variability in its outputs, the application-specific correctness of LLM text generation remains largely unexplored.

*Therefore, in this chapter, we propose a method to bridge this gap for URL classification self-explanations. As our one-shot classification framework outputs both explanation and prediction, we assess the correctness of the URL benign and phishing indicators identified in the self-explanations by comparing them to post-hoc explanations obtained through a supervised training setting.*

## 5.3 Our framework

As mentioned in the introduction, our aim is to address the problem of generalisability and explainability in phishing URL detection through a one-shot LLM-based classification framework. We implement a lightweight approach that leverages LLMs' vast embedded knowledge to improve classification accuracy while reducing false positive rates. This approach also provides explanations for each prediction to improve usability and empower end-users with more knowledge about a given URL prediction.

Our framework consists of prompting the LLM with the instruction and one example, as shown in Figure 5.1. Following the prompt we provide the URL for which we require an output from the LLM. The objective of this prompt is to not only obtain a single-word prediction but also obtain the reasoning or explanation that leads to the prediction. In particular, we prompt a targeted form of LLM reasoning by specifically instructing the LLM

to consider the benign and phishing characteristics of the URL. This form of targeted reasoning taps into LLM knowledge regarding legitimate entities and websites present on the web. It also encourages the LLM to consider the structure of the URL and whether it reflects phishing characteristics.

In finalising our framework we faced the following technical challenges:

- **Prompting for classification:** To query an LLM about whether a given URL is benign or phishing we experimented with two variations of prompting. The first type gave the LLM the following four options to choose from when deciding a classification - benign/phishing/potentially phishing/uncertain. This was motivated by the idea that allowing more granular labelling options would drive the LLM to make more precise decisions. We processed the LLMs' outputs to this prompt by considering all benign labelled URLs as benign and the rest as phishing. That is, it is safer to consider a URL as phishing if there is any uncertainty or potential for harm. The second type of prompting we tried is the one used in our proposed framework. Our initial analysis showed that the second prompt style returned slightly higher accuracy than the first prompt. The reason for this is that the availability of the 'potentially phishing' and 'uncertain' label options encouraged the LLM to take a cautious approach during classification. As a result, many benign URLs with less well-known domains were labelled as 'uncertain' or 'potentially phishing', thereby increasing the false positive rate. In contrast, the second prompt type (our proposed framework) encouraged the LLM to be more decisive about benign URL domains which improved the overall classification accuracy.
- **Leveraging reasoning capabilities:** Inspired by chain-of-thought (CoT) reasoning, some of our initial experiments verified that prompting the LLM to reason before making a prediction improved detection accuracy. To implement this in our prompt, we experimented with two versions of prompting. The first version followed CoT methods more closely and prompted the LLM to consider the domain, subdomain, and path of the URL separately and identify benign and phishing characteristics before making a prediction. The second version simply prompted the LLM to consider the benign and phishing characteristics (this is the version used in our final proposed method). With the first prompt version, we found that explicitly identifying benign and phishing characteristics in the various URL components caused the LLM to place unnecessary weight on these features which resulted in lower overall prediction accuracy. Instead allowing the LLM more flexibility in its reasoning by simply prompting it to consider the general benign and phishing characteristics of the URL (second prompt version) resulted in higher overall detection accuracy.

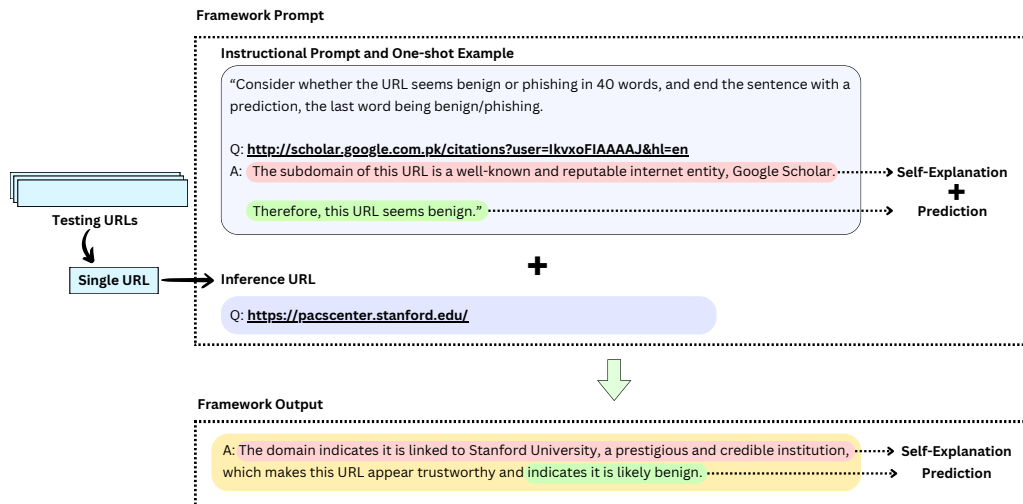


Figure 5.2: LLM-based one-shot URL classification framework. Our framework produces natural language explanations for each prediction.

- Number of examples to include in prompt:** We experimented with varying numbers of few-shot examples to assess if increasing the number of examples improved the prediction accuracy. Besides the one-shot approach, we experimented with two examples (one phishing URL and one benign URL) and five examples (three benign URLs and two phishing URLs). Our analysis showed that increasing the number of examples did not improve the prediction accuracy by a significant amount and therefore decided on a one-shot approach.

Figure 5.2 illustrates the overall flow of our framework. In particular, we show how we consider the first part of the output to be a *self-explanation* for the prediction that follows it. In the following section, we evaluate the performance of five large language models in returning accurate predictions for URLs in such a one-shot setting. We also evaluate the quality of the self-explanations that accompany the prediction.

Here, note that we operate in a one-shot setting, i.e., we use only one URL (i.e., the Google Scholar URL as the sole example) to generate all of our results in Section 5.5. We emphasise that our prompt is the same for all of our results (i.e., it is the same query with the same Google Scholar URL). Later in Section 5.5.1, we show the viability of our approach in zero-shot and few-shot settings as well.

## 5.4 Experiment settings

In this section, we first introduce the datasets we use to evaluate the prediction accuracy of our framework. Next, we describe how we evaluate the prediction performance of LLMs

as one-shot URL classifiers in comparison to supervised URL classifiers, followed by the experiment settings we use to evaluate the quality of LLM self-explanations.

### 5.4.1 Datasets

We use the ISCX-2016 [45], EBBU [4] and HP [187] datasets similar to the previous chapters and consider the same training, validation and testing splits as Table 3.1. To evaluate our LLM-based framework, we use a randomly sampled subset of 1,000 URLs from the test splits of each dataset. We limit our analysis to 1,000 samples due to cost considerations when accessing some of the LLM APIs.

### 5.4.2 Accuracy comparison with supervised URL classifiers

We evaluate the prediction performance of LLMs under our framework by comparing their accuracy to state-of-the-art URL classifiers trained in a fully supervised setting. For this, we refer to Chapter 3 where we trained four state-of-the-art URL classifiers: URLNet [1], URLTran [2], CatchPhish [3] and PhishRF [4]. We trained and tested the performance of these classifiers using the training and testing datasets described in Table 3.1. We presented the performance of the supervised URL classifiers when applied to each test dataset in Table 3.2. Here, note that the models work well when they are tested on the test set which comes from the same dataset. For example, the URLTran model trained on the ISCX dataset achieves an F1 score of 0.99 when tested on the ISCX test set. However, its performance drops significantly to F1 scores of 0.69 and 0.68 when tested on the EBBU and HP test sets, respectively.

We obtain the one-shot prediction accuracies under our framework using the 1,000 randomly selected samples of the test datasets in Table 3.1 as described in Section 5.4.1. That is, in each query, we prompted the LLMs to reason and predict the label of one given URL from the test sets. As our framework specifically instructed for the prediction to be stated in the final word of the output as either ‘benign’ or ‘phishing’ we considered each output prediction to be either benign or phishing if the last 20 characters of the output contained the terms ‘benign’ and ‘phishing’ respectively. If the last 20 characters contained neither of the terms we considered the prediction to be ‘uncertain’. However, in the calculation of accuracy, we counted the ‘uncertain’ predictions as phishing, considering that would be the safer decision in a practical setting.

We tested five state-of-the-art LLMs - GPT 4-Turbo [94], Claude 3 Opus [207], Gemini [208], LLaMA 3 [209], and LLaMA 2 [95]. We repeated our experiments for each LLM and each test dataset five times and calculated the average F1 score. We select the F1 score

as a performance metric as it accounts for the true positive rate as well as the false positive rate. We compare the F1 scores of each LLM and each test set under our framework with the F1 scores obtained by supervised URL classifiers in Table 3.2. Since LLMs can generate different outputs to the same query at different times, for each of the 1,000 URLs, we queried the LLM five times and reported the average performance results together with the standard deviation. We present these results in Section 5.5.1.

### 5.4.3 Quality evaluation of LLM self-explanations

We evaluate the quality of the LLM self-explanations from two aspects. First, we compare the benign and phishing indicators given by LLMs with the indicators returned by a post-hoc explanation method (i.e., LIME [210]) applied to the corresponding supervised URL classifier. This is to justify further that the few-shot predictions the LLMs make are logical and often have the same reasoning as the corresponding supervised learning classifier that has trained on large volumes of training data. Second, we evaluate the general quality of the LLM-generated explanations in terms of readability, coherence, and informativeness using the G-Eval [199] framework.

#### 5.4.3.1 Alignment between LLM and LIME indicators

Classifying a given URL as benign or phishing is an overall decision made by the classifier observing various indicators of a given URL. The datasets in this area (**cf.** Table 3.1) consist of only URLs and their corresponding labels. Therefore, there is no gold-standard ground truth to the explanations provided by the LLM on benign and phishing indicators.

As a result, we establish a proxy metric by comparing the LLMs' self-explanations to the indicators returned by a post-hoc explanation method applied to a supervised URL classifier. Specifically, we select URLTran as it returned the highest average F1 score across the three datasets (**cf.** Table 3.2) in the supervised setting. We applied the LIME algorithm [211] to three separate URLTran models trained on each of the three training sets and obtained LIME indicators for the models' predictions for their respective test sets.

The LIME algorithm determines feature attributions of a given test sample by approximating a sparse linear model based on the model predictions for inputs sampled from the neighbourhood of that test sample. Feature importance is then determined by the feature coefficients of that linear model. As each URLTran model returned F1 scores of 0.99 for their respective test sets, we assume that the sparse linear model learned by LIME and, therefore, the indicators it returned are a true reflection of the benign and phishing characteristics represented within labelled URL datasets.

By comparing the LLM self-explanations with the LIME indicators, we can verify how well benign and phishing characteristics identified by LLMs align with label-specific features learnt by supervised models from URL-labelled datasets. We measure this alignment using Jaccard similarity.

Specifically, we queried each LLM with the prompt in Figure 5.3. We followed this prompt with the URLs from our test set and their corresponding self-explanations from the respective LLMs, one at a time, to obtain the self-explanation indicators. For each URL, we compared the self-explanation indicators with the LIME indicators that were returned for the same URL from the corresponding URLTran model. For example, if the given URL belonged to the HP test set, we compared the LLM self-explanation indicators to the LIME indicators for the same URL when applied to URLTran trained on the HP training set.

#### LLM self-explanation Example

“You are an intelligent assistant. For the given URL, return only the specific URL parts that the explanation considers benign or phishing. You must use the format ‘Benign:Phishing:’.”

##### Q. URL:

`http://www.vivscreisveci.vcirveseiaveesi.ghqphy.top/uWBRvZ8quj/page1.php`

**Exp:** The URL contains a suspect domain that is not recognizable and appears to have randomly generated parts, which is typical of phishing sites aiming to mimic legitimate ones deceptively. This URL seems to be phishing.

**A. Benign:** ‘ ‘ **Phishing:** ‘vivscreisveci.vcirveseiaveesi.ghqphy.top’, ‘uWBRvZ8quj’

##### Q. URL:

`https://reciclatex.com/ES/cx/home`

**Exp:** This URL seems suspicious because it lacks recognizable branding and could be mimicking a legitimate site with its structure, but without contextual evidence of malicious intent, it’s uncertain. My prediction leans towards benign.

**A. Benign:** ‘com’, ‘ES’, ‘https’ **Phishing:** ‘reciclatex’, ‘home’, ‘cx’

Figure 5.3: Prompting the LLM to list benign and phishing indicators identified in the self-explanation

**Jaccard similarity:** To numerically compare the two sets of indicators (i.e., LLMs and LIME), we used the Jaccard similarity coefficient [212], which calculates a similarity score between two sets as the ratio of the intersection of the sets and their union:  $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$ . Here, the label-specific LIME indicators form one set of elements and the indicators identified by the LLM self-explanation form another set. As each self-explanation was in

natural language, we extracted the specific URL components that it identified using LLM prompting.

**Example Jaccard similarity calculation:**

- URL: `https://reciclatex.com/ES/cx/home`
- Explanation: “This URL seems suspicious because it lacks recognizable branding and could be mimicking a legitimate site with its structure, but without contextual evidence of malicious intent, it’s uncertain. My prediction leans towards benign.”
- Self-explanation indicators: *Benign*: ‘com’, ‘ES’, ‘https’ *Phishing*: ‘reciclatex’, ‘home’, ‘cx’

As the original LLM output predicted the URL as benign, we collect the benign self-explanation indicators delimited by special characters (A: ‘com’, ‘ES’, ‘https’) and compare them with corresponding benign LIME indicators (B: ‘com’, ‘ES’, ‘reciclatex’, ‘https’, ‘home’). Using the Jaccard similarity measure, this corresponds to:

$$J(A, B) = \frac{|\{\text{'com'}, \text{'ES'}, \text{'https'}\}|}{|\{\text{'com'}, \text{'ES'}, \text{'https'}, \text{'home'}, \text{'reciclatex'}\}|} = 0.6. \quad (5.1)$$

The Jaccard similarity score ranges from 0 to 1, where higher similarity scores indicate a greater overlap between self-explanation indicators and LIME indicators. We conduct the above evaluation for each of the LLMs that we evaluate under our framework and for each of the three datasets. We report these results in Section 5.5.2.1.

### 5.4.3.2 G-Eval framework

To evaluate the general quality of LLM outputs, we use the G-Eval framework [199], which implements LLMs to evaluate the quality of a text. Specifically, the framework uses a chain-of-thought approach to use an LLM to evaluate a text against a given metric. In order to be useful to the end-users of a URL classification system, we assess the self-explanations according to the following metrics: *readability*, *coherence*, and *informativeness*. We adopt the definition of ‘readability’ provided in [213] for cyber security question-answering tasks, and we adapt the definitions of ‘coherence’ and ‘informativeness’ from [214] to suit our application as described below.

- **Readability (1-5):** Assesses how easily the average reader comprehends a text, considering factors such as lexical, syntactic, semantic, and stylistic complexity.

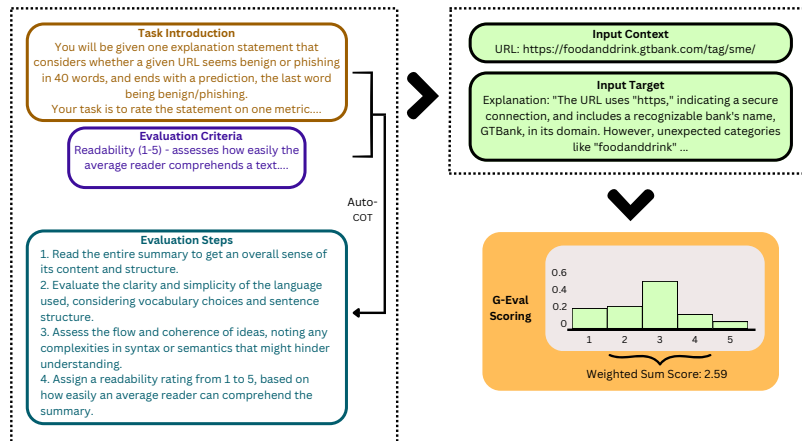


Figure 5.4: Using the G-Eval framework to assess the quality of LLM self-explanations

- **Coherence (1-5):** Assesses the collective quality of all sentences. This metric measures how well the statement is structured and organised in explaining why the URL is predicted to be either benign or phishing.
- **Informativeness (1-5):** Measures how well the output answers the question. That is, this metric assesses how clearly the statement considers the benign and phishing characteristics of the URL and finally provides a prediction.

In the G-Eval framework, an LLM evaluates a given text against a given criteria and the criteria definition by following a sequence of evaluation steps generated by the LLM itself through chain-of-thought reasoning. That is, an LLM is given a criterion and its definition and prompted to produce a series of steps that it should follow to evaluate a text against that criterion for a specific task. As we are investigating five LLMs for prediction accuracy and explanation quality under our one-shot framework, we use a different LLM (GPT-4o) in our G-Eval implementation to evaluate the other five LLMs' self-explanations.

Figure 5.4 which is adapted from the original paper [199] illustrates our use of G-Eval for the readability metric. We give the *task introduction* and the *evaluation criteria* to GPT-4o to generate evaluation steps. Then we input each test URL (given as context) and the corresponding self-explanation (the text to be evaluated), and the G-Eval framework calculates a probability-weighted summation of all possible scores (1-5). We followed a similar approach for the coherency and informativeness metrics.

We generate readability, coherence, and informativeness scores for the self-explanations corresponding to all URLs from the three test sets generated by the five LLMs that we evaluate under our one-shot framework. We report these results in Section 5.5.2.2.

Table 5.2: Prediction performance of one-shot classification - F1 Scores. GPT 4-Turbo demonstrates the highest average F1 scores across all three datasets with an overall average of 0.92.

Test Dataset	Models					
	URLTran (Supervised)	GPT 4	Claude 3	Gemini	LLaMA 3	LLaMA 2
HP	0.99	<b>0.94</b> ±0.00153	0.89±0.00092	0.87±0.00529	0.84±0.00358	0.67±0.00215
EBBU	0.99	<b>0.95</b> ±0.00393	0.91±0.00184	0.86±0.01191	0.87±0.00378	0.69±0.00220
ISCX	0.99	<b>0.87</b> ±0.00471	0.85±0.00284	0.78±0.00589	0.81±0.00348	0.67±0.00202

## 5.5 Results

### 5.5.1 Prediction performance of one-shot LLM URL classifiers

In Table 5.2, we present the F1 scores of our one-shot URL classification framework on the five LLMs we test (cf. Section 5.4.2): GPT 4-Turbo, Claude 3 Opus, Gemini, LLaMA 3, and LLaMA 2. As explained in Section 5.4.2, these results are the average F1 score calculated over five repeats of the experiment. For ease of comparison, we also present the F1 scores obtained by URLTran (i.e., best-performing model under the supervised setting - cf. Table 3.2) when applied to each test set after being trained on the respective training set.

We observe that among the LLMs, GPT 4-Turbo produces the highest average F1 scores across all three datasets, with an overall average of 0.92. This is only 0.07 less than the performance of the fully supervised learned classifier. Claude demonstrates the next best performance with an overall average of 0.88, while Gemini and LLaMA 3 perform similarly, with overall averages of 0.83 and 0.84, respectively. Meanwhile, LLaMA 2 returns the lowest overall average F1 score of 0.68. The standard deviations over five repeats are very low for all the LLMs portraying overall consistency in accuracy over the five independent runs.

A more detailed analysis of the LLM outputs revealed that one of the reasons for LLaMA 2’s significantly low performance may be its inability to generate good self-explanations. For many of the URLs, LLaMA 2 returned a prediction without considering the benign and phishing characteristics of the URL, as our prompt had asked. Furthermore, even when the LLMs did consider the benign and phishing characteristics of the URL, in comparison to the other LLMs, GPT 4-Turbo returned more accurate predictions. Consider the following LLM outputs for the benign URL <https://reconciliation.americanexpress.com/>.

- **GPT 4-Turbo:** *“The domain hints at a well-known financial institution, American Express, but the use of ”reconciliation” might raise suspicions without further context. Despite potential concerns, the domain’s legitimacy suggests the URL is likely benign.”*

- **Claude 3:** *“The URL appears to be a legitimate American Express subdomain for account reconciliation. However, without further context, it’s best to be cautious. Visit the main American Express website directly to avoid potential phishing attempts. The URL seems benign, but caution is advised.”*
- **Gemini:** *“The subdomain of this URL is reconciliation which is not a common subdomain of the American Express website. Therefore, this URL seems phishing.”*
- **LLaMA 3:** *“The URL appears to be a legitimate domain from a well-known financial institution, American Express. However, the specific page seems to be a secure login page, which may raise some concerns.”*
- **LLaMA 2:** *“The subdomain of this URL is American Express, which has been associated with phishing attacks in the past. Therefore, this URL seems phishing.”*

From the above examples, we observe that all LLMs are able to correctly identify the main entity in the URL domain as well as the context of the entity (financial institution). However, knowledge of the fact that many phishing attacks are financial attacks leads LLaMA 2 to directly classify the URL as phishing, which is an incorrect prediction. Meanwhile, Gemini incorrectly decides that the subdomain is not a common or expected one for the American Express website, which also leads it to predict the URL as phishing. While Claude 3 and LLaMA 3 correctly acknowledge that the domain and subdomain are legitimate, similar to LLaMA 2 and Gemini, the risk posed by the financial context of the URL, as well as the connection to account login/reconciliation, prevents the LLMs from classifying the URL as benign. In contrast to all the above, GPT 4-Turbo acknowledges the potential for suspicion given both of the above factors but proceeds to predict the URL as benign with apparent confidence (“likely benign”).

Another example demonstrates that LLaMA 3 and LLaMA 2 do not follow the correct reasoning based on their knowledge of legitimate entities, and this leads to an incorrect prediction. The benign URL `https://drfone.wondershare.net/ad/` returned the following outputs from each LLM:

- **GPT 4-Turbo:** *“The URL directs to a specific page of Wondershare’s drfone, a recognized software for phone data management, implying it’s a promotional link from a legitimate company. Consequently, this URL appears benign.”*
- **Claude 3:** *“The URL is for a software product called Dr.Fone by Wondershare, which appears to be a legitimate data recovery tool. The .net top-level domain and lack of suspicious elements suggest this is likely a benign URL for the product’s website or an advertisement. Benign.”*

- **Gemini:** *“This URL belongs to a software company and therefore seems benign.”*
- **LLaMA 3:** *“The URL appears to be a promotional link from a software company, Wondershare, which may be trying to entice users to download their product. Therefore, this URL seems phishing.”*
- **LLaMA 2:** *“A: The subdomain of this URL is not a well-known or reputable internet entity, which suggests that it may be a phishing website. Prediction: Phishing.”*

Again, we observe that GPT 4-Turbo recognises the domain and subdomain as legitimate and, therefore, classifies the URL as benign. Claude 3 and Gemini outputs follow a similar pattern. LLaMA 2, however, does not correctly recognise the *Wondershare* entity, while for LLaMA 3, the possibility of the URL being a ‘promotional link’ leads it to predict the URL as phishing.

Overall, the results show that the LLMs’ self-explanation capabilities have an effect on the prediction performance. Specific to the phishing domains, these reasoning capabilities involve correctly identifying legitimate entities in the domains and subdomains of the URLs, as well as correctly attributing phishing and benign characteristics to the URLs. For example, despite correctly stating American Express has been associated with phishing attacks (as phishing URLs often include well-known entities in their URLs), LLaMA 2 failed to correctly identify the legitimacy of the specific American Express URL in the example above.

Finally, we highlight the low standard deviation values of the LLM F1 scores in Table 5.2. This shows that LLMs are pretty consistent with their predictions over the five experiment runs, something we further analyse in Section 5.6.3. We also provide the accuracy, precision and recall results of this experiment, as well as the comparison of one-shot classification with supervised models trained on combinations of our three training sets in Table A.1 and Table A.2 in the Appendix.

## 5.5.2 Quality of LLM outputs

### 5.5.2.1 Alignment between LLM self-explanations and LIME indicators - Jaccard similarity

As mentioned before, the purpose of this evaluation is to justify further that the one-shot predictions the LLMs make are logical and often have the same reasoning as the corresponding supervised learning classifier that has trained on large volumes of training data. From our repeated experiments to assess the prediction performance of our one-shot classification framework, we obtained five outputs for each URL from each of the five LLMs. Therefore,

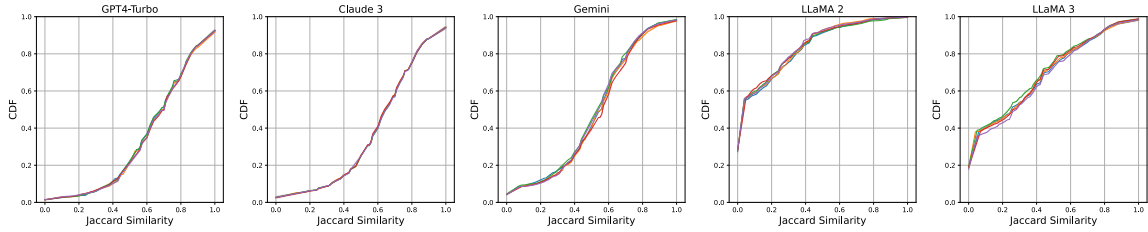


Figure 5.5: Cumulative distribution of the Jaccard similarity between LIME and LLM indicators - HP Dataset. The CDFs for GPT 4-Turbo and Claude 3 follow a strong sigmoid trend indicating high alignment between LIME and LLM indicators.

to evaluate the self-explanations produced by our framework, for each URL, we applied our Jaccard similarity evaluation to each of the five repeats for each URL. In Figure 5.5, we present the cumulative distributive functions (CDF) of the Jaccard similarity scores obtained for each LLM applied to the HP dataset. Each curve in each figure represents the CDF of the Jaccard Similarity values obtained from a single run. For brevity, we present results only for the HP dataset, and the other two datasets had similar patterns as the HP dataset. We observe that the CDFs for GPT 4-Turbo and Claude 3 follow a strong sigmoid trend for each URL dataset, indicating they are distributed more towards high values (i.e., high alignment between LIME and LLM indicators).

More concretely, for each URL, we calculate the mean Jaccard similarity score to reflect the overall alignment between its self-explanation indicators and LIME indicators. In Table 5.3 we show the proportion of the Jaccard scores that were above 0.5 for each dataset and each model. Here, we find that for GPT 4-Turbo, 72%, 69%, and 56% of the Jaccard similarity scores were above 0.5 for the HP, EBBU, and ISCX test sets, respectively. These proportions gradually decrease for Claude 3, Gemini and LLaMA 3. For LLaMA 2 we find the lowest proportions, reflecting that very few of its LLM self-explanations were aligned with the LIME indicators.

Table 5.3: Percentage of Jaccard similarity scores  $\geq 0.5$ . GPT 4-Turbo reflects the highest alignment between its self-explanations and LIME indicators as the proportion of Jaccard scores above 0.5 are all above 50%. This alignment gradually decreases for the other models.

Data	LLM				
	GPT 4	Claude 3	Gemini	LLaMA3	LLaMA2
HP	72%	69%	51%	22%	6%
EBBU	69%	65%	32%	29%	8%
ISCX	56%	48%	23%	19%	7%

To further exemplify this, in Table 5.4 we show a case where for a given URL, GPT 4-Turbo and LLaMA 2 return contradicting predictions and self-explanations. For each self-explanation, we used the prompt in Figure 5.3 to extract the self-explanation indicators.

Table 5.4: Examples of alignment between LLM self-explanation indicators and LIME indicators

URL: <a href="https://drfone.wondershare.net/ad/">https://drfone.wondershare.net/ad/</a> / Original label: Benign				
LLM	Self-Explanation	Self-Explanation Indicators	LIME indicators	Jaccard Score
GPT 4-Turbo	"The URL directs to a specific page of Wondershare's dr.fone, a recognized software for phone data management, implying it's a promotional link from a legitimate company. Consequently, this URL appears benign."	'drfone.wondershare.net', 'ad', 'https'	'wondershare', 'net', 'https', 'ad'	0.8
LLaMA 2	"The subdomain of this URL is not a well-known or reputable internet entity, which suggests that it may be a phishing website. Prediction: Phishing."	'drfone', 'wondershare', 'ad'	'drfone'	0.3

We then compared these indicators to the LIME indicators corresponding to the LLM prediction. Therefore, we compare the GPT self-explanation indicators with the LIME benign indicators and the LLaMA 2 self-explanation indicators with the LIME phishing indicators. We observe that regardless of the predictions returned by each LLM, GPT 4-Turbo identifies the label-specific indicators more accurately than LLaMA 2, which is expressed by the Jaccard similarity between the respective self-explanation indicators and the LIME indicators.

*Overall, these results indicate that the self-explanations returned by GPT 4-Turbo and Claude 3 are more consistent with the benign and phishing characteristics represented in URL datasets. As observed in the examples shown in the previous section, this is likely due to GPT 4-Turbo and Claude 3's ability to describe specific URL components as benign and phishing indicators when reasoning the predictions that both make.*

### 5.5.2.2 G-Eval analysis

In Figure 5.6, we present the probability density functions (PDF) of the scores returned for each metric, LLM and dataset by the G-Eval framework for the HP dataset. For brevity, we present only the results for the HP dataset and the results for the other datasets had similar patterns. It can be observed that for all LLMs apart from LLaMA 2, much of the probability mass lies above 0.8, indicating high readability, coherency, and informativeness in most of the self-explanations.

Observing the individual LLM outputs revealed that in many cases, unlike the other LLMs, LLaMA 2 failed to provide an explanation altogether. For example, for the URL <http://scholar.google.com.ua/citations?user=r7GEXWwAAAAJ&hl=ru>, for three out of the five repeats of our evaluation, the output from LLaMA 2 was:

"Sure, here are my predictions:

Q: <http://scholar.google.com.pk/citations?user=IkvxofIAAAAJ&hl=en> - Benign

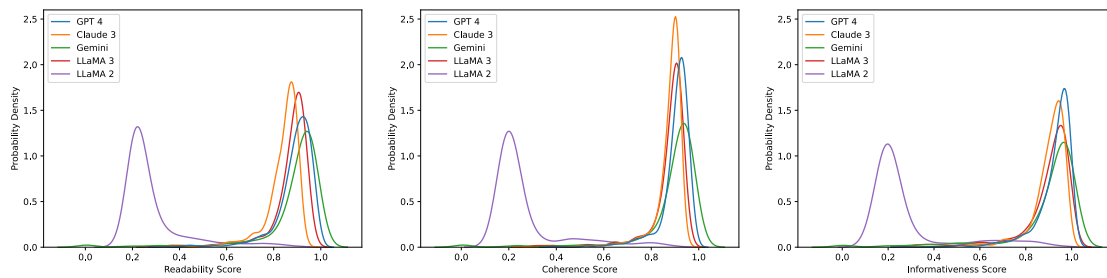


Figure 5.6: Distribution of G-Eval scores for readability, coherence and informativeness for the HP dataset

*Q: <http://scholar.google.com.ua/citations?user=r7GEXWwAAAAJ&hl=ru> - Phishing”*

This self-explanation results in a readability score of only 0.22 as the output not only lacks a self-explanation but also includes the prediction of the example URL given in the original prompt, making the readability of this output unnecessarily complex for a potential end-user. For coherence, this output obtained a score of 0.21, again due to the lack of detailed explanations and logical flow that led to the prediction. In comparison, Gemini and GPT 4-Turbo obtained coherence scores of 0.64 and 0.87, respectively. Their respective outputs for the above URL are given below.

- **Gemini:** *“This URL leads to a cited Google Scholar page with an author’s publication data, making it appear benign.”*
- **GPT 4-Turbo:** *“Similar to the previous URL, this one also resides under the Google Scholar subdomain, pointing to a reputable source. It appears to be a user profile page on Google Scholar, making it seem benign.”*

It is clear from the above that the high coherence scores for Gemini and GPT 4-Turbo result from the logical flow demonstrated in the self-explanations above. However, the coherence of the GPT 4 output is strengthened with the additional note that ‘Google Scholar’ is a ‘reputable source’. Such information is vital for improving user awareness and training for phishing detection.

While coherence assessed the structure and logical flow of the self-explanations, informativeness assessed how well the statements answered the original prompt and considered the benign and phishing characteristics of the URL. For instance, LLaMA 2 and GPT 4-Turbo returned the following explanations for the URL <https://pizza.dominos.com/missouri/hollister/>

- **LLaMA 2:** *“The subdomain of this URL is a well-known pizza delivery chain, Domino’s Pizza. Therefore, this URL seems benign. Prediction: The URL will direct to a legitimate Domino’s Pizza website.”*
- **GPT 4-Turbo:** *“The URL points to a specific Domino’s pizza location in Missouri, using a domain structure consistent with the official Domino’s site, indicating that it is likely a legitimate link. This URL seems benign.”*

For the above outputs, LLaMA 2 and GPT 4-Turbo obtained informativeness scores of 0.52 and 0.98, respectively. While LLaMA 2 scored low for failing to follow the specific instruction of ending the output with either ‘benign’ or ‘phishing’, GPT 4-Turbo scored higher for characterising additional URL components, such as mentioning Missouri to be a specific location.

*Overall, this analysis shows that the top-performing LLM outputs are useful for the end-users of a phishing detection application. The three metrics we analysed here assessed the ease with which a user can comprehend the text (readability), the logical flow of arguments (coherence) and how well the text answered the question while identifying benign and phishing characteristics (informativeness). Besides LLaMA 2, the remaining four LLMs generally scored well in terms of all metrics, demonstrating a high overall quality of explanation. However, additionally accounting for GPT 4-Turbo’s superior prediction accuracy, we can conclude that GPT 4-Turbo is the best overall performing model within our framework.*

## 5.6 Extended analysis

In this section, we present an extended analysis of the various aspects of our framework.

### 5.6.1 Zero-shot and five-shot prompting

First, we investigate to what extent including the example in the query helps the final performance. To this end, we conduct experiments for zero-shot and five-shot settings. To evaluate our framework under a zero-shot approach, we repeated the same approach as in Section 5.4.2 without any examples in the prompt. For the five-shot approach, we generated self-explanations and predictions for five URLs by prompting instructions shown in Figure 5.2 to GPT 4o (i.e., a different LLM than the ones we evaluate). We randomly selected these five URLs from the original training datasets. Below, we list the five URLs and their corresponding self-explanations and predictions.

**Benign:**

- <https://www.youtube.com/premium> (Source: HP): Given that the URL points to YouTube’s official domain and a known service, it appears to be a legitimate and safe link. Prediction: benign.
- <http://www.dictionary.com/browse/lan> (Source: EBBU): The URL belongs to the reputable website Dictionary.com and leads to a specific word page, making it likely trustworthy and legitimate. Prediction: benign.
- [http://allrecipes.com/Recipe/Midwest-Salisbury-Steak/Detail.aspx?soid=recs\\_recipe\\_9](http://allrecipes.com/Recipe/Midwest-Salisbury-Steak/Detail.aspx?soid=recs_recipe_9) (Source: ISCX): The URL appears to link to a legitimate page on AllRecipes, a well-known cooking site that usually includes similar syntax in its links. Therefore, this URL seems benign.

**Phishing:**

- <http://marlianstv.com/loan/office365/> (HP): The URL includes recognizable terms but the domain "marlianstv.com" does not match Microsoft or Office365’s official sites, which raises suspicion. Therefore, this URL seems phishing.
- <http://fb.manage-pages.com/> (Source: EBBU): The domain in this URL does not belong to the official Facebook site, making it likely to be a phishing attempt. Prediction: phishing.

For our zero-shot analysis, we repeated the experiment five times as was done in Section 5.4.2; however, for the five-shot approach, we conducted the experiment only once. This is because of the cost considerations of the higher context window size of the five-shot setting.

We report the results in Table 5.5. We can observe overall that GPT4-Turbo remains the best-performing LLM in both the zero-shot and five-shot settings, and its average F1 score across the three datasets is 0.92, which is the same as the one-shot performance (cf. Table 5.2). The performance of the other LLMs seems to vary in the zero-shot and five-shot settings. For example, Claude 3’s F1 scores are 0.91, 0.88, and 0.87 for zero-shot, one-shot, and five-shot settings, respectively. The corresponding values for Gemini are 0.86, 0.84, and 0.84.

Meanwhile, LLaMA 3 and LLaMA 2 experience the greatest increases. The average F1 scores for Llama 3 are 0.84, 0.84, and 0.89 for zero-shot, one-shot, and five-shot settings. For LLaMA 2, the corresponding numbers are 0.48, 0.68, and 0.80.

*In summary, although increasing the number of examples in the prompt improves performance for some models, the top-performing GPT-4 Turbo remains unaffected by the number*

Table 5.5: Prediction performance - F1 Score of zero-shot and five-shot settings. The results reflect that increasing the number of examples in the prompt improves performance for some models. However, the top-performing GPT 4-Turbo remains unaffected by the number of examples.

Test Dataset	Zero-shot					
	URLTran (Supervised)	GPT 4	Claude 3	Gemini	LLaMA 3	LLaMA 2
HP	0.99	0.95±0.00529	0.94±0.00125	0.91±0.00362	0.91±0.00321	0.52±0.00601
EBBU	0.99	0.92±0.00620	0.94±0.00114	0.87±0.00405	0.83±0.00557	0.46±0.00930
ISCX	0.99	0.88±0.00624	0.85±0.00241	0.81±0.01109	0.77±0.00873	0.45±0.01087
Test Dataset	Five-shot					
	URLTran (Supervised)	GPT 4	Claude 3	Gemini	LLaMA 3	LLaMA 2
HP	0.99	0.94	0.88	0.90	0.90	0.82
EBBU	0.99	0.94	0.92	0.88	0.92	0.85
ISCX	0.99	0.88	0.86	0.81	0.86	0.74

of examples. This finding is significant because it shows that solutions using our framework can save costs by employing zero or one-shot settings, reducing the number of querying tokens without compromising performance.

## 5.6.2 Increasing the training data size in the supervised setting

Next, we investigate whether we can train the supervised training models using less training data and how the one-shot results stand compared to those. To this end, we progressively train the URLTran models using 10%, 20%, and so on, up to 100% of the training data. We show these results in Figure 5.7. In the plots, the solid lines represent the supervised learning performance, and the dashed lines represent the one-shot learning performance in a given dataset.

For each training dataset, we observe that there is an increase in F1 scores of the same test set as the training set size increases, and the scores are usually above 0.95. For example, when training on the HP training data and testing on the HP data, the F1 scores increase from 0.98 to 0.99. The corresponding observation for the EBBU dataset shows an increase from

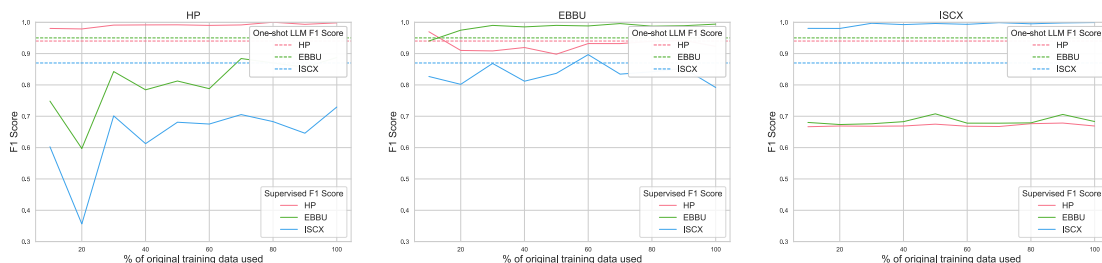


Figure 5.7: Change in F1 scores while increasing the training set size. We observe that in supervised settings, cross-dataset performance is poor at lower training set sizes while our one-shot framework consistently yields higher performance.

0.94 to 0.99. Meanwhile, the performance of URLTran on the other test sets demonstrates fluctuations.

Nonetheless, the advantage of our method is clearly visible in the HP dataset. While the same dataset test performance is high for this, the cross-dataset performances are particularly low when the HP-trained model is tested on EBBU and ISCX datasets, especially at lower training set sizes. For example, when trained only using 20% of data of HP, the URLTran model achieves 0.6 and 0.35 F1 scores on EBBU and ISCX test sets, respectively. Meanwhile, the performance obtained using our one-shot framework was 0.94, 0.95 and 0.87, respectively.

*These results indicate that while URLTran models can be trained with fewer data, their cross-dataset performance remains problematic, particularly at lower training set sizes. In contrast, our one-shot framework delivers higher and more consistent performance across all three datasets using only one training sample.*

### 5.6.3 Prediction consistency

Accounting for the probabilistic nature of large language models, we additionally analysed the variability of predictions for a given URL. For this, we analysed the predictions returned by LLMs in each of the five repeats in the one-shot setting. For each URL, we collected the predictions returned by each LLM and calculated the Gini Impurity index.

Generally used for the decision tree algorithm [215, 216], the Gini Impurity index measures the variety of types of elements in a dataset using the following formula:  $Gini(d) = 1 - \sum_{n=1}^N p_n^2$ . Here,  $d$  represents the dataset,  $n$  represents a type of element,  $N$  represents the total number of different types and  $p_n$  represents the probability that type  $n$  occurs in  $d$ . A lower Gini impurity index reflects a higher purity or a lower variability of types within a given dataset.

For our prediction consistency scenario, a lower Gini Impurity, therefore, translates to higher consistency in predictions. Consider the example of a URL that receives the following set of predictions from an LLM:  $\{Benign, Benign, Phishing, Benign, Benign\}$ . For this set of elements, the Gini Impurity is  $1 - \left(\frac{4}{5}\right)^2 - \left(\frac{1}{5}\right)^2 = 0.32$ . In contrast, a URL that receives the same prediction across all five runs (i.e., all benign or all phishing) will have a Gini Impurity of  $1 - \left(\frac{5}{5}\right)^2 = 0$ .

In Figure 5.8, we present the Gini Impurity distributions obtained for URLs from each test dataset from each LLM. Note that our processing of LLM outputs resulted in three possible prediction outcomes; ‘benign’, ‘phishing’, and ‘uncertain’ (cf. Section 5.4.2). This results in five possible Gini Impurity scores: 0, 0.32, 0.48, 0.56, 0.64.

We observe from Figure 5.8 that the majority of Gini Impurity scores for all LLMs and

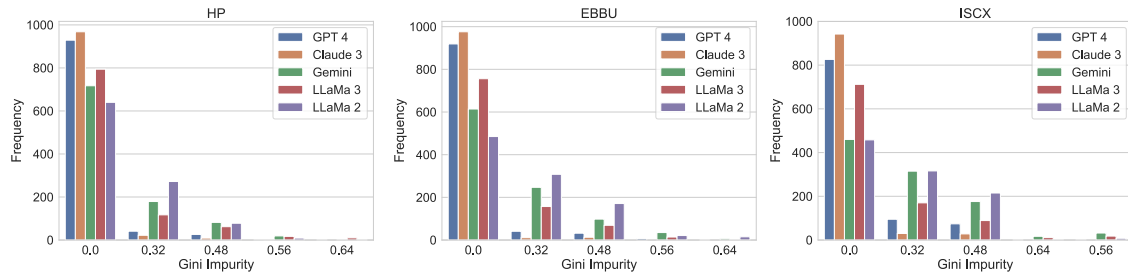


Figure 5.8: Histograms of Gini impurity values for assessing prediction consistency. As all models reflect high frequency for low impurity scores, the results reflect that LLMs provide consistent predictions for a given URL.

all test datasets are 0. This reflects that most of the URLs obtained completely consistent predictions. We further analysed how many cases comprised of URLs consistently receiving an ‘uncertain’ prediction; that is, the proportion of cases where the prediction was deemed ‘uncertain’ for all five repeats. We found that for GPT 4-Turbo, there were no cases across the three datasets where the prediction was ‘uncertain’ for all five repeats and only 25 cases in total where at least one prediction out of the five was ‘uncertain’. In comparison, Claude returned 50 cases across the three datasets where at least one prediction out of the five was ‘uncertain’. This count increased for Gemini, LLaMA 3 and LLaMA 2, with each returning 412, 249 and 1,300, respectively. Nonetheless, the number of cases where the prediction was consistently ‘uncertain’ was only 4, 25, 6 and 9 for Claude, Gemini, LLaMA 3 and LLaMA 2, respectively. These results again reinforce the overall superior quality of the GPT 4-Turbo outputs. Among the five LLMs, GPT 4-Turbo returned the lowest number of cases where at least one prediction was uncertain and no cases where the prediction was consistently ‘uncertain’.

In terms of returning completely consistent predictions (for each URL all five predictions are the same; Gini impurity of 0), GPT 4 and Claude demonstrate the best performance with Claude demonstrating slightly higher consistency than GPT 4. However, considering that Claude’s overall prediction accuracy was lower than GPT 4, this reflects that some of Claude’s predictions were consistently incorrect.

*In summary, this result demonstrates that, despite being probabilistic generative models, top-performing LLMs provide consistent predictions for a given URL, making them ideal for practical URL classification systems.*

### 5.6.4 Where do LLMs perform better?

Next, we take a closer look at the specific cases where our one-shot approach outperforms supervised training and vice versa. That is, we analyse the URLs that GPT 4-Turbo predicts correctly under our one-shot approach while the supervised URLTran mispredicts, and vice versa. As shown in Table 5.6, the number of cases where GPT 4-Turbo mispredicts a URL while URLTran predicts it correctly is significantly less than the number of cases where URLTran mispredicts a URL while GPT 4-Turbo predicts correctly. For example, the number of benign URLs from the HP dataset that GPT 4-Turbo mispredicts and URLTran predicts correctly is only 31. In contrast, the number of benign URLs from the HP dataset that GPT 4-Turbo predicts correctly while URLTran mispredicts is only 228. This is the case for both benign and phishing URLs, reflecting that GPT 4-Turbo returns both fewer false positives and fewer false negatives. We note that this result highlights the benefits of leveraging LLMs’ knowledge of benign websites. As most LLMs are trained on textual data from the internet, LLMs’ background knowledge on benign sites enables the significant reduction in false positive rates through our one-shot framework. Overall, our work demonstrates that the one-shot LLM framework provides better generalisation ability than supervised models trained on large URL datasets.

Analysing the self-explanations of GPT 4-Turbo’s false positives and false negatives, we find that, in general, a benign URL is classified as phishing when the domain of the URL is not widely reputable or the structure of the URL looks suspicious. Similarly, the self-explanations for the false negative cases reflected that GPT 4-Turbo recognised the URL domain as legitimate. We verified that for some of the false negative cases, the self-explanation of GPT 4-Turbo was correct. That is, the URL was indeed hosted on a legitimate site. However, the URL was labelled as phishing as the content of the webpage contained phishing elements. This is, in any case, a limitation of phasing URL classifiers, as we discuss later.

Models		HP		EBBU		ISCX	
URLTran	GPT 4	B	P	B	P	B	P
✓	✓	214	242	230	237	198	206
✓	✗	31	3	19	6	41	32
✗	✓	228	250	235	246	219	245
✗	✗	27	5	16	11	42	17

Table 5.6: Cases of similar and dissimilar predictions from URLTran and GPT 4-Turbo under our one-shot framework

## 5.7 Discussion

We demonstrated that LLMs can be effectively used for phishing URL classification with a one-shot prompt, achieving performance comparable to supervised models. Additionally, they overcome the cross-dataset performance limitations seen in state-of-the-art URL classifiers. Next, we discuss the implications and limitations of our work, as well as potential future directions.

**Use of URL features only:** Our classification approach focused solely on using URL features. This method has several advantages, including the ability to detect zero-day phishing campaigns and eliminating the need to visit the actual website or rely on external sources for predictions. Moreover, this method is applicable when handling URL-shortening and URL-redirect scenarios, as an additional step can be taken to resolve the real URL via a DNS lookup before making a prediction. However, a comprehensive approach to phishing URL detection would also incorporate other features such as landing page content, IP address and domain registration information, along with known blacklisted and whitelisted URLs. This is because, on some occasions, the URL may not contain the information required to make the prediction. Nonetheless, with widespread deployment, including the integration of LLMs into corporate product suites and locally deployed LLM models [217, 218], our approach can serve as one front in the fight against phishing attempts. Simply generating a warning and explanation on end-user devices could potentially save thousands, if not millions, of potential phishing victims.

**Multi-modal LLMs:** Increasingly, LLMs are becoming multi-modal [219]. In our context, vision-language models that take both text and images as inputs and provide text outputs are particularly relevant. Examples of such models include BLIP-2 [220] and LLaVA [221]. A promising future direction is to utilise these models to include web page content in few-shot LLM decision-making. Such methods could be further enhanced by incorporating webpage code, leveraging the LLMs' ability to interpret code [222] to improve upon previous similar attempts using more conventional deep learning methods [223, 224]. However, the downside of this approach is the necessity to visit the web page, take a screenshot using a headless browser, save the code, and query the LLM. This process introduces latency and is more suitable for a server-side solution where these actions occur before the emails with links are made accessible to the end user.

**Assessing the quality of LLM explanations:** Awareness and user behaviour are crucial in the fight against phishing. As mentioned in the introduction, previous research has shown that providing users with simple warnings and explanations helps them make more informed

decisions [188, 189]. In this work, we used the automated G-Eval framework to measure the readability, coherence, and informativeness of LLM-generated self-explanations about URLs. We defined these metrics specifically for our problem by referring to existing literature. Another approach could be to conduct a user study to measure these aspects and gather further feedback on the quality of the explanations.

In the absence of annotated data with benign and phishing indicators, our explainability analysis used LIME indicators as the gold standard for explanations. Despite the supervised classifiers achieving near-perfect accuracy on their test sets, there is a possibility that LIME explanations may not always be accurate. This is a challenging issue because any explainable AI method will have this limitation. One potential solution is to use an expertly annotated dataset with explanations for various phishing and benign indicators. However, creating such a dataset with the help of cybersecurity experts would be costly.

## 5.8 Conclusion

In this work, we propose an LLM-based one-shot learning framework for malicious URL classification. Using three separate URL datasets and five state-of-the-art LLMs, we evaluate the performance of our framework in terms of prediction performance as well as the quality of LLM self-explanations. Our experiments demonstrate that our one-shot framework performs similarly to completely supervised URL classifiers. Specifically, GPT 4-Turbo is found to be the best-performing LLM under our one-shot framework. Its prediction performance across the three datasets is within 0.05 – 0.12 F1 score of a state-of-the-art supervised URL classifier (URLTran), with an overall average F1 score of 0.92. We also quantitatively demonstrated that the LLM self-explanations align with post-hoc explanations of the supervised classifiers and, using the G-Eval framework, demonstrate their high readability, coherency and informativeness. Future work in this area can include implementing multi-modal LLMs to incorporate a greater variety of features in the classification process going beyond URL features and incorporating DNS/whois information, source codes, and visual features of the web page.

# Chapter 6

## Conclusion

By applying recent developments in deep learning, statistical analyses and thorough evaluations, this thesis contributes to understanding how to mitigate domain shift for privacy-preserving and robust phishing URL detection.

First, we investigated the transfer performance of existing machine learning and deep learning-based phishing detection methods and the domain shift difficulties arising from different sources of URLs. This analysis highlighted the nature of distributional differences among URL datasets contributing to noise and model overfitting and thereby causing low transfer performance on URLs of diverse sources. Incorporating these findings, we proposed a federated model that overcomes URL data heterogeneity among clients and achieves high phishing detection performance while maintaining client data privacy.

Furthermore, to address the problem of false positives in phishing detection, we leveraged the knowledge of pretrained large language models (LLMs). We investigated few-shot prompting strategies and proposed a one-shot framework that achieved detection performance comparable to a completely supervised model. The proposed method also employed LLM reasoning capabilities and provided natural language explanations for each URL classification, thereby improving the usability of the classifier.

In this chapter, we provide an overview of the key findings and contributions from the previous chapters, which focused on enhancing phishing detection from the perspective of domain shift mitigation. The rest of the chapter is structured as follows. Section 6.1 provides a summary of each chapter, emphasising the key research ideas and contributions along with their limitations. Finally, Section 6.2 concludes the chapter by outlining potential directions for future research.

## 6.1 Summary and conclusion

### 6.1.1 Phishing URL detection using Unsupervised Domain Adaptation

In Chapter 3, we proposed an Unsupervised Domain Adaptation (UDA) framework to improve the generalisability of phishing URL classifiers. Our work investigated existing state-of-the-art classifiers and demonstrated their poor transfer performance on URLs with feature distributions varying from the samples used to train the detection models. We implemented statistical tests to conduct comprehensive analyses of the phishing URL classifiers and different URL datasets to reveal domain shift problems in phishing URLs collected from different sources. More specifically, our findings revealed the specific features with distribution shifts between the source (training) and target (test) datasets which caused reduced transfer performance on phishing URL classifiers.

Based on our findings, we proposed a UDA framework that aligned these specific features across the source and target URL datasets. We evaluated our feature-alignment framework on four state-of-the-art classifiers using three URL datasets and demonstrated an average F1 score increase of 0.06 and for some scenarios the increase was as high as 0.2. The results indicated that our method can meaningfully enhance model transferability, enabling more effective threat intelligence sharing between organisations. Overall, with this work we aimed to contribute towards building robust phishing detection models trained on diverse URL datasets.

While our approach provided useful insights, one of the limitations was that we performed individual feature analysis in isolation. That is, our analysis did not account for inter-feature correlations and the effect of one feature-alignment on another feature. This resulted in reduced transfer performance in some cases (cf. Section 3.8.4).

Another key constraint arose from the nature of UDA which assumed an entirely unlabelled target domain. Consequently, our framework aligned overall feature distributions without considering class-specific distributions which reduced the effectiveness of our solution for specific models. While accessing some labelled data would be useful to inform the training process, we note that accessing diverse phishing URL datasets is difficult due to data privacy considerations. We addressed this issue in our work on federated phishing detection.

Overall, this chapter improved understanding of domain shift in phishing URLs. Our UDA framework provided the best results with the URLTran model reflecting the superiority of transformer-based models. This suggested the potential of achieving even greater improvements from large language models which we explored in a later chapter.

### 6.1.2 Federated phishing detection with FeUDA-BERT

In Chapter 4, we presented FeUDA-UBERT, a federated, domain-aware BERT-based model for privacy-preserving phishing URL detection. The model introduced a URL domain-aware self-attention mechanism that mitigated client-side data heterogeneity by emphasising the relative importance of tokens based on their position. This component ensures higher importance is assigned to tokens in the URL domain while other tokens are given comparatively lower importance. We also introduced a self-attention-based client selection strategy inspired by curriculum learning. By clustering clients' self-attention weights, this step ensured that at each communication round, the model parameter aggregation occurred between similar clients.

We evaluated our contributions both individually and in combination. Our analysis found that both contributions resulted in improvements over baseline models, and when combined, they achieved an average F1 score of 0.93 on test data. Similar to the previous chapter, we also conducted statistical analyses assessing the statistical significance of our results. We applied the Wilcoxon signed-rank test to paired results comparing baseline models with different combinations of our proposed contributions and the results consistently demonstrated that the improvements due to our solution are statistically significant.

One of the limitations of our solution is related to the URL domain-aware self-attention component. Here, we established a strict definition of the target self-attention map by pre-defining the peak position of the importance distribution. An improvement to this method would be to make the target self-attention map learnable. That is, the key and query matrices learned during training can be regularised to follow a certain pattern (such as giving higher importance to domain tokens) instead of being strictly defined (cf. Section 4.3.1). However, for this to yield a robust federated phishing detection would require a client local data population that collectively represents diverse URLs.

### 6.1.3 One-shot phishing URL detection with large language models

In Chapter 5, we proposed a one-shot LLM-based framework for phishing URL detection. Our framework leveraged the general knowledge of benign entities and websites embedded in pretrained language models to address the problem of false positives in phishing detection. Furthermore, we accounted for the usability of the model by exploiting LLMs' reasoning capabilities and produced easy-to-understand explanations for URL classification.

We evaluated our approach on five state-of-the-art LLMs using three distinct URL datasets, measuring both prediction performance and quality of explanations. Our results showed that

the framework performed comparably to fully supervised classifiers trained on large annotated datasets. Notably, GPT-4 Turbo achieved an average F1 score of 0.92, which was within 0.05–0.12 of the fully supervised URLTran classifier. Furthermore, using the G-Eval framework, we assessed the readability, coherence, and informativeness of LLM-generated self-explanations which aligned well with post-hoc explanations from traditional models.

This one-shot approach demonstrated that LLMs can generalise well across URL datasets — addressing a key limitation in existing phishing classifiers as explored in Chapter 3. Moreover, the method eliminates the need for labelled data, while generating an understandable explanation for predictions. However, one of the limitations identified in this work was that the LLMs largely relied on knowledge of well-reputed benign entities and websites in making predictions. While this reduced false positive rates, the remaining misclassifications were due to a lack of knowledge on less well-known legitimate sites, as was apparent based on the explanations provided with each classification (**cf.** Section 5.6.4). This limitation can be addressed through a federated system that enables privacy-preserving data sharing to build a well-informed phishing detection model. We discuss this and other possible future work below.

## 6.2 Future work

We now explore potential research directions informed by the findings of this thesis.

### 6.2.1 Combining URL domain shift mitigation with diverse features

One of the key features of this thesis is the focus on URLs only for phishing detection. As mentioned previously, this approach has various advantages from avoiding aversion techniques such as cloaking and URL-shortening to detecting zero-day phishing campaigns. In practical scenarios, however, our findings from this thesis can be combined with other data types to enhance phishing detection even further. For instance, several existing works in deep learning combine URL-based features with a variety of others such as features derived from DNS, hosting features, webpage HTML features, and third-party features [182, 225, 19]. However, these do not consider domain shift mitigation for enhancing detection performance which is a research direction that remains to be explored.

Similarly, in Chapter 5, we touched on the potential of multi-modal transformer models for phishing detection. Studies applying this in other contexts include [226] and [227], where transformer models combining text and image are applied to hate speech detection and image understanding tasks, respectively. In voice phishing detection, [228] introduces a

multimodal transformer model integrating call transcripts, sequences of mobile screenshots, and extracted Optical Character Recognition (OCR) text.

A very recent and relevant work in this regard is [229], which develops a multimodal LLM agent that combines webpage features, such as logo images, along with HTML information and URL features for phishing detection. The method further leverages online and offline knowledge bases to extract well-known brand entities and their associated information. Again, however, similar to the existing work analysed in this thesis, the transfer performance of the method is not evaluated.

It would be interesting, therefore, to incorporate our findings regarding URL domain shift into more recent techniques similar to those mentioned above. Doing so would likely reveal further areas requiring investigation, such as domain shifts in webpage features, which can be addressed using domain adaptation methods such as the framework we proposed in Chapter 3. However, as these methods rely on knowledge of the target domain, which is often inaccessible, a privacy-preserving method like the federated solution proposed in Chapter 4 may be required. We discuss potential research direction in that area below.

## 6.2.2 Combining federated phishing detection and LLMs

One of the main themes addressed in this thesis is the difficulty of obtaining highly representative datasets to enable robust phishing URL detection model training. Due to the fast-paced and dynamic nature of phishing campaigns and the reluctance of organisations to share their private threat intelligence data, phishing URL detection models are adversely affected by domain shifts, as discussed above. In Chapter 4, we presented a privacy-preserving solution to intelligence sharing while overcoming client-level data heterogeneity problems. Our work demonstrated the importance of overcoming URL-based heterogeneity in a federated setting for phishing detection.

We noted in that chapter that our URL domain-aware self-attention component that overcomes URL data heterogeneity can be improved by making it trainable with certain regularisation parameters, but that a well-representative client data population would be required for this to be effective. That is, the robustness of a phishing detection model learned in a federated setting largely depends on the diversity of data represented by the clients' local data collectively. In other words, the generalisability of a federated phishing detection model is limited to the data of its clients, who may represent organisations of niche sectors and, therefore, lack representation of generally well-known benign URLs.

On the other hand, our work in Chapter 5 demonstrated how large language models can be employed as one-shot phishing URL classifiers to yield results comparable to supervised models. More importantly, we extract natural language explanations from LLMs which

reflect the vast amount of general knowledge embedded in pretrained LLMs that enables high detection performance. In particular, we observe that large language models depend highly on the knowledge of well-reputed legitimate sites to decide whether a given URL is malicious or benign. On the other hand, we observed LLMs to be limited by their lack of knowledge of newer, less-known benign websites, leading to false positives.

Based on the above, we suggest future work that combines a federated solution such as FeUDA-BERT with LLMs. This would aim to leverage the general knowledge of well-reputed legitimate sites from LLMs along with knowledge of niche benign websites coming from the federated model clients to overcome the issue of false positives in phishing detection.

Other studies combining federated learning and LLMs include [230, 231]. In [232], the authors studied the personalisation versus robustness tradeoffs of using federated prompt tuning for large language models. Su et al. [230], on the other hand, investigated federated prompt tuning that is adaptive to multi-domain settings for collaborative image classification. However, the combination of federated approaches with LLMs for phishing detection remains largely unexplored, specifically accounting for URL-based heterogeneity among clients.

# Bibliography

- [1] 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.
- [2] Pranav Maneriker, Jack W Stokes, Edir Garcia Lazo, Diana Carutasu, Farid Tajaddodianfar, and Arun Gururajan, “URLTran: Improving Phishing URL Detection Using Transformers,” in *Proceedings of MILCOM 2021-2021 IEEE Military Communications Conference (MILCOM)*. IEEE, 2021, pp. 197–204.
- [3] Routhu Srinivasa Rao, Tatti Vaishnavi, and Alwyn Roshan Pais, “CatchPhish: detection of phishing websites by inspecting URLs,” *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, pp. 813–825, 2020.
- [4] Ozgur Koray Sahingoz, Ebubekir Buber, Onder Demir, and Banu Diri, “Machine learning based phishing detection from URLs,” *Expert Systems with Applications*, vol. 117, pp. 345–357, 2019.
- [5] Australian Competition Consumer Commission, “Australian Competition Consumer Commission Annual Report 2024,” <https://www.accc.gov.au/system/files/annual-report-2023-24.pdf>, [Accessed 09-02-2025].
- [6] Internet Crime Complaint Center, “Federal Bureau of Investigation Internet Crime Report 2023,” [https://www.ic3.gov/AnnualReport/Reports/2023\\_ic3report.pdf](https://www.ic3.gov/AnnualReport/Reports/2023_ic3report.pdf), [Accessed 09-02-2025].
- [7] FBI National Press Office, “FBI Announces Nationwide ‘Take A Beat’ Campaign to Increase Awareness of Frauds and Scams,” <https://www.fbi.gov/news/press-releases/fbi-announces-nationwide-take-a-beat-campaign-to-increase-awareness> [Accessed 09-02-2025].

- [8] IBM, “IBM X-Force Threat Intelligence Index 2024,” <https://www.ibm.com/reports/threat-intelligence>, [Accessed 09-02-2025].
- [9] Min Wu, Robert C Miller, and Simson L Garfinkel, “Do security toolbars actually prevent phishing attacks?,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2006, pp. 601–610.
- [10] Elena Zheleva, Aleksander Kolcz, and Lise Getoor, “Trusting spam reporters: A reporter-based reputation system for email filtering,” *ACM Transactions on Information Systems (TOIS)*, vol. 27, no. 1, pp. 1–27, 2008.
- [11] Justin Ma, Lawrence K Saul, Stefan Savage, and Geoffrey M Voelker, “Identifying suspicious URLs: an application of large-scale online learning,” in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 681–688.
- [12] Davide Canali, Marco Cova, Giovanni Vigna, and Christopher Kruegel, “Prophiler: a fast filter for the large-scale detection of malicious web pages,” in *Proceedings of the 20th International Conference on World Wide Web*, 2011, pp. 197–206.
- [13] Guang Xiang, Jason Hong, Carolyn P Rose, and Lorrie Cranor, “Cantina+ a feature-rich machine learning framework for detecting phishing web sites,” *ACM Transactions on Information and System Security (TISSEC)*, vol. 14, no. 2, pp. 1–28, 2011.
- [14] Choon Lin Tan, Kang Leng Chiew, Kelvin SC Yong, Johari Abdullah, Yakub Sebastian, et al., “A graph-theoretic approach for the detection of phishing webpages,” *Computers & Security*, vol. 95, pp. 101793, 2020.
- [15] Tao Feng and Chuan Yue, “Visualizing and interpreting rnn models in url-based phishing detection,” in *Proceedings of the 25th ACM Symposium on Access Control Models and Technologies*, 2020, pp. 13–24.
- [16] Yongjie Huang, Qiping Yang, Jinghui Qin, and Wushao Wen, “Phishing URL detection via CNN and attention-based hierarchical RNN,” in *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE, 2019, pp. 112–119.
- [17] Alper Ozcan, Cagatay Catal, Emrah Donmez, and Behcet Senturk, “A hybrid DNN–LSTM model for detecting phishing URLs,” *Neural Computing and Applications*, pp. 1–17, 2023.

- [18] Tommy Chin, Kaiqi Xiong, and Chengbin Hu, “Phishlimiter: A phishing detection and mitigation approach using software-defined networking,” *IEEE Access*, vol. 6, pp. 42516–42531, 2018.
- [19] Mohammed M Alani and Hissam Tawfik, “PhishNot: a cloud-based machine-learning approach to phishing URL detection,” *Computer Networks*, vol. 218, pp. 109407, 2022.
- [20] Adam Oest, Penghui Zhang, Brad Wardman, Eric Nunes, Jakub Burgis, Ali Zand, Kurt Thomas, Adam Doupé, and Gail-Joon Ahn, “Sunrise to sunset: Analyzing the end-to-end life cycle and effectiveness of phishing attacks at scale,” in *Proceedings of 29th USENIX Security Symposium (USENIX Security 20)*, 2020.
- [21] Kaspersky, “Powerful but short-lived: one third of phishing pages cease to be active after a day,” <https://www.kaspersky.com/about/press-releases/powerful-but-short-lived-one-third-of-phishing-pages-cess-to-be-a> [Accessed 09-02-2025].
- [22] Adam Oest, Yeganeh Safaei, Penghui Zhang, Brad Wardman, Kevin Tyers, Yan Shoshitaishvili, and Adam Doupé, “{PhishTime}: Continuous Longitudinal Measurement of the Effectiveness of Anti-phishing Blacklists,” in *Proceedings of 29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 379–396.
- [23] Jing Zhang, Wanqing Li, and Philip Ogunbona, “Joint geometrical and statistical alignment for visual domain adaptation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1859–1867.
- [24] Jing Jiang and ChengXiang Zhai, “Instance weighting for domain adaptation in NLP,” *ACL*, 2007.
- [25] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith, “Don’t stop pretraining: Adapt language models to domains and tasks,” *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.
- [26] Bushra A Alahmadi, Louise Axon, and Ivan Martinovic, “99% false positives: A qualitative study of {SOC} analysts’ perspectives on security alarms,” in *Proceedings of 31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 2783–2800.
- [27] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen, “TRANCO: A Research-Oriented Top Sites Ranking

- Hardened Against Manipulation,” *Proceedings of Network and Distributed Systems Security (NDSS) Symposium 2019*, 2019.
- [28] Mona Mojdeh and Gordon V Cormack, “A Mail Client Plugin for Privacy-Preserving Spam Filter Evaluation.,” in *CEAS*, 2008.
- [29] Keno Albrecht, Nicolas Burri, and Roger Wattenhofer, “Spamato-An Extendable Spam Filter System.,” in *CEAS*. Citeseer, 2005.
- [30] Ian Fette, Norman Sadeh, and Anthony Tomasic, “Learning to detect phishing emails,” in *Proceedings of the 16th International Conference on World Wide Web*, 2007, pp. 649–656.
- [31] Niels Provos Panayiotis Mavrommatis and MARF Monrose, “All your iFRAMES point to us,” in *Proceedings of the 17th conference on Security symposium*. USENIX Association Berkeley, 2008, pp. 1–16.
- [32] Yue Zhang, Jason I Hong, and Lorrie F Cranor, “Cantina: a content-based approach to detecting phishing web sites,” in *Proceedings of the 16th International Conference on World Wide Web*, 2007, pp. 639–648.
- [33] Saeed Abu-Nimeh, Dario Nappa, Xinlei Wang, and Suku Nair, “A comparison of machine learning techniques for phishing detection,” in *Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit*, 2007, pp. 60–69.
- [34] 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 *Proceedings of 30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 3793–3810.
- [35] Ankit Kumar Jain and Brij B Gupta, “A machine learning based approach for phishing detection using hyperlinks information,” *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, pp. 2015–2028, 2019.
- [36] Kang Leng Chiew, Choon Lin Tan, KokSheik Wong, Kelvin SC Yong, and Wei King Tiong, “A new hybrid ensemble feature selection framework for machine learning-based phishing detection system,” *Information Sciences*, vol. 484, pp. 153–166, 2019.
- [37] Ravindu De Silva, Mohamed Nabeel, Charith Elvitigala, Issa Khalil, Ting Yu, and Chamath Keppitiyagama, “Compromised or Attacker-Owned: A large scale classification and study of hosting domains of malicious URLs,” in *Proceedings of 30th USENIX security symposium (USENIX security 21)*, 2021, pp. 3721–3738.

- [38] Cheng Huang, Shuang Hao, Luca Invernizzi, Jiayong Liu, Yong Fang, Christopher Kruegel, and Giovanni Vigna, “Gossip: Automatically identifying malicious domains from mailing list discussions,” in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, 2017, pp. 494–505.
- [39] Pratyusa Manadhata, Sandeep Yadav, Prasad Rao, and William Horne, “Detecting malicious domains via graph inference,” in *Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop*, 2014, pp. 59–60.
- [40] Sandeep Yadav, Ashwath Kumar Krishna Reddy, AL Narasimha Reddy, and Supranamaya Ranjan, “Detecting algorithmically generated malicious domain names,” in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, 2010, pp. 48–61.
- [41] Leyla Bilge, Engin Kirda, Christopher Kruegel, and Marco Balduzzi, “EXPOSURE: Finding Malicious Domains Using Passive DNS Analysis.,” in *Proceedings of Network and Distributed System Security (NDSS) Symposium 2011*, 2011, pp. 1–17.
- [42] Sujata Garera, Niels Provos, Monica Chew, and Aviel D Rubin, “A framework for detection and measurement of phishing attacks,” in *Proceedings of the 2007 ACM workshop on Recurring malware*, 2007, pp. 1–8.
- [43] D. Kevin McGrath and Minaxi Gupta, “Behind Phishing: An Examination of Phisher Modi Operandi,” in *Proceedings of USENIX Workshop on Large-Scale Exploits and Emergent Threats*, 2008.
- [44] Justin Ma, Lawrence K Saul, Stefan Savage, and Geoffrey M Voelker, “Beyond blacklists: learning to detect malicious web sites from suspicious URLs,” in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009, pp. 1245–1254.
- [45] Mohammad Saiful Islam Mamun, Mohammad Ahmad Rathore, Arash Habibi Lashkari, Natalia Stakhanova, and Ali A Ghorbani, “Detecting malicious urls using lexical analysis,” in *Proceedings of 10th International Conference on Network and System Security, NSS 2016*. Springer, 2016, pp. 467–482.
- [46] Huaping Yuan, Zhenguo Yang, Xu Chen, Yukun Li, and Wenyin Liu, “URL2Vec: URL modeling with character embeddings for fast and accurate phishing website detection,” in *Proceedings of IEEE Intl Conference on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud*

- Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*. IEEE, 2018, pp. 265–272.
- [47] Tomáš Šipka, Milan Šulc, and Jiří Matas, “The hitchhiker’s guide to prior-shift adaptation,” in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2022, pp. 1516–1524.
- [48] Nathalie Japkowicz and Shaju Stephen, “The class imbalance problem: A systematic study,” *Intelligent data analysis*, vol. 6, no. 5, pp. 429–449, 2002.
- [49] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer, “SMOTE: synthetic minority over-sampling technique,” *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.
- [50] Jingbo Zhu and Eduard Hovy, “Active learning for word sense disambiguation with methods for addressing the class imbalance problem,” in *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 2007, pp. 783–790.
- [51] Yee Seng Chan and Hwee Tou Ng, “Word Sense Disambiguation with Distribution Estimation.,” in *Proceedings of the 19th International Joint Conference on Artificial intelligence*, 2005, vol. 5, pp. 1010–5.
- [52] Hidetoshi Shimodaira, “Improving predictive inference under covariate shift by weighting the log-likelihood function,” *Journal of statistical planning and inference*, vol. 90, no. 2, pp. 227–244, 2000.
- [53] Bianca Zadrozny, “Learning and evaluating classifiers under sample selection bias,” in *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 114.
- [54] Jiayuan Huang, Arthur Gretton, Karsten Borgwardt, Bernhard Schölkopf, and Alex Smola, “Correcting sample selection bias by unlabeled data,” *Proceedings of Advances in Neural Information Processing Systems*, vol. 19, 2006.
- [55] Steffen Bickel and Tobias Scheffer, “Dirichlet-enhanced spam filtering based on biased samples,” *Proceedings of the 20th International Conference on Neural Information Processing Systems*, vol. 19, 2006.
- [56] Zak Murez, Soheil Kolouri, David Kriegman, Ravi Ramamoorthi, and Kyungnam Kim, “Image to image translation for domain adaptation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4500–4509.

- [57] Safa Cicek and Stefano Soatto, “Unsupervised domain adaptation via regularized conditional alignment,” in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 1416–1425.
- [58] Zihan Liu, Yan Xu, Tiezheng Yu, Wenliang Dai, Ziwei Ji, Samuel Cahyawijaya, Andrea Madotto, and Pascale Fung, “CrossNER: Evaluating cross-domain named entity recognition,” in *Proceedings of the AAAI conference on Artificial Intelligence*, 2021, vol. 35, pp. 13452–13460.
- [59] Sining Sun, Binbin Zhang, Lei Xie, and Yanning Zhang, “An unsupervised deep domain adaptation approach for robust speech recognition,” *Neurocomputing*, vol. 257, pp. 79–87, 2017.
- [60] Wei-Ning Hsu, Yu Zhang, and James Glass, “Unsupervised domain adaptation for robust speech recognition via variational autoencoder-based data augmentation,” in *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2017, pp. 16–23.
- [61] Seyed Ali Rokni and Hassan Ghasemzadeh, “Synchronous dynamic view learning: a framework for autonomous training of activity recognition models using wearable sensors,” in *Proceedings of the 16th ACM/IEEE International Conference on Information Processing in Sensor Networks*, 2017, pp. 79–90.
- [62] Md Abdullah Al Hafiz Khan, Nirmalya Roy, and Archan Misra, “Scaling human activity recognition via deep learning-based domain adaptation,” in *Proceedings of 2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE, 2018, pp. 1–9.
- [63] Xiaohang Xu, Hao Peng, Md Zakirul Alam Bhuiyan, Zhifeng Hao, Lianzhong Liu, Lichao Sun, and Lifang He, “Privacy-preserving federated depression detection from multisource mobile health data,” *IEEE Transactions on Industrial Informatics*, vol. 18, no. 7, pp. 4788–4797, 2021.
- [64] Yejin Kim, Jimeng Sun, Hwanjo Yu, and Xiaoqian Jiang, “Federated tensor factorization for computational phenotyping,” in *Proceedings of the 23rd ACM SIGKDD International conference on knowledge discovery and data mining*, 2017, pp. 887–895.
- [65] Arthur Jochems, Timo M Deist, Issam El Naqa, Marc Kessler, Chuck Mayo, Jackson Reeves, Shruti Jolly, Martha Matuszak, Randall Ten Haken, Johan van Soest,

- et al., “Developing and validating a survival prediction model for NSCLC patients through distributed learning across 3 countries,” *International Journal of Radiation Oncology\* Biology\* Physics*, vol. 99, no. 2, pp. 344–352, 2017.
- [66] Rajesh Kumar, Abdullah Aman Khan, Jay Kumar, Noorbakhsh Amiri Golilarz, Simin Zhang, Yang Ting, Chengyu Zheng, Wenyong Wang, et al., “Blockchain-federated-learning and deep learning models for covid-19 detection using ct imaging,” *IEEE Sensors Journal*, vol. 21, no. 14, pp. 16301–16314, 2021.
- [67] Dianbo Liu, Dmitriy Dligach, and Timothy Miller, “Two-stage federated phenotyping and patient representation learning,” in *Proceedings of the conference. Association for Computational Linguistics. Meeting*, 2019, vol. 2019, p. 283.
- [68] Sijing Duan, Deyu Zhang, Yanbo Wang, Lingxiang Li, and Yaoxue Zhang, “JointRec: A deep-learning-based joint cloud video recommendation framework for mobile IoT,” *IEEE Internet of Things Journal*, vol. 7, no. 3, pp. 1655–1666, 2019.
- [69] Alberto Caballero, Teresa García-Valverde, Fernando Pereñíguez, and Juan A Botía, “Activity recommendation in intelligent campus environments based on the Eduroam federation,” *Journal of ambient intelligence and smart environments*, vol. 8, no. 1, pp. 35–46, 2016.
- [70] Omar Abdel Wahab, Gaith Rjoub, Jamal Bentahar, and Robin Cohen, “Federated against the cold: A trust-based federated learning approach to counter the cold start problem in recommendation systems,” *Information Sciences*, vol. 601, pp. 189–206, 2022.
- [71] Ben Tan, Bo Liu, Vincent Zheng, and Qiang Yang, “A federated recommender system for online services,” in *Proceedings of the 14th ACM conference on recommender systems*, 2020, pp. 579–581.
- [72] David Neumann, Andreas Lutz, Karsten Müller, and Wojciech Samek, “A privacy preserving system for movie recommendations using federated learning,” *ACM Transactions on Recommender Systems*, vol. 3, no. 2, pp. 1–51, 2024.
- [73] Khalil Muhammad, Qinqin Wang, Diarmuid O’Reilly-Morgan, Elias Tragos, Barry Smyth, Neil Hurley, James Geraci, and Aonghus Lawlor, “Fedfast: Going beyond average for faster training of federated recommender systems,” in *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 2020, pp. 1234–1242.

- [74] Xiaoming Yuan, Jiahui Chen, Jiayu Yang, Ning Zhang, Tingting Yang, Tao Han, and Amir Taherkordi, “Fedstn: Graph representation driven federated learning for edge computing enabled urban traffic flow prediction,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 8, pp. 8738–8748, 2022.
- [75] Zhaoyang Du, Celimuge Wu, Tsutomu Yoshinaga, Kok-Lim Alvin Yau, Yusheng Ji, and Jie Li, “Federated learning for vehicular internet of things: Recent advances and open issues,” *IEEE Open Journal of the Computer Society*, vol. 1, pp. 45–61, 2020.
- [76] Mustain Billah, Sk Tanzir Mehedi, Adnan Anwar, Ziaur Rahman, and Rafiqul Islam, “A systematic literature review on blockchain enabled federated learning framework for internet of vehicles,” *arXiv preprint arXiv:2203.05192*, 2022.
- [77] Yiqiang Chen, Xin Qin, Jindong Wang, Chaohui Yu, and Wen Gao, “Fedhealth: A federated transfer learning framework for wearable healthcare,” *IEEE Intelligent Systems*, vol. 35, no. 4, pp. 83–93, 2020.
- [78] Binhang Yuan, Song Ge, and Wenhui Xing, “A federated learning framework for healthcare IoT devices,” *arXiv preprint arXiv:2005.05083*, 2020.
- [79] Dianbo Liu, Timothy Miller, Raheel Sayeed, and Kenneth D Mandl, “FADL: Federated-autonomous deep learning for distributed electronic health record,” *Proceedings of Machine Learning for Health (ML4H) Workshop at NeurIPS 2018*, 2018.
- [80] Zhongyun Tang, Haiyang Hu, and Chonghuan Xu, “A federated learning method for network intrusion detection,” *Concurrency and Computation: Practice and Experience*, vol. 34, no. 10, pp. e6812, 2022.
- [81] Ruijie Zhao, Yue Yin, Yong Shi, and Zhi Xue, “Intelligent intrusion detection based on federated learning aided long short-term memory,” *Physical Communication*, vol. 42, pp. 101157, 2020.
- [82] Hong Liu, Shuaipeng Zhang, Pengfei Zhang, Xinqiang Zhou, Xuebin Shao, Geguang Pu, and Yan Zhang, “Blockchain and federated learning for collaborative intrusion detection in vehicular edge computing,” *IEEE Transactions on Vehicular Technology*, vol. 70, no. 6, pp. 6073–6084, 2021.
- [83] Vishnu Pandi Chellapandi, Liangqi Yuan, Christopher G Brinton, Stanislaw H Żak, and Ziran Wang, “Federated learning for connected and automated vehicles: A survey of existing approaches and challenges,” *IEEE Transactions on Intelligent Vehicles*, vol. 9, no. 1, pp. 119–137, 2023.

- [84] Manoj Basnet and Mohd Hasan Ali, “A deep learning perspective on connected automated vehicle (CAV) cybersecurity and threat intelligence,” in *Deep learning and its applications for Vehicle Networks*, pp. 39–56. CRC Press, 2023.
- [85] Beibei Li, Yuhao Wu, Jiarui Song, Rongxing Lu, Tao Li, and Liang Zhao, “DeepFed: Federated deep learning for intrusion detection in industrial cyber–physical systems,” *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5615–5624, 2020.
- [86] Virraaji Mothukuri, Prachi Khare, Reza M Parizi, Seyedamin Pouriyeh, Ali Dehghan-tanha, and Gautam Srivastava, “Federated-learning-based anomaly detection for IoT security attacks,” *IEEE Internet of Things Journal*, vol. 9, no. 4, pp. 2545–2554, 2021.
- [87] Bimal Ghimire and Danda B Rawat, “Recent advances on federated learning for cybersecurity and cybersecurity for federated learning for internet of things,” *IEEE Internet of Things Journal*, vol. 9, no. 11, pp. 8229–8249, 2022.
- [88] Anran Li, Lan Zhang, Juntao Tan, Yaxuan Qin, Junhao Wang, and Xiang-Yang Li, “Sample-level data selection for federated learning,” in *Proceedings of IEEE INFO-COM 2021-IEEE Conference on Computer Communications*. IEEE, 2021, pp. 1–10.
- [89] Eunjeong Jeong, Seungeun Oh, Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim, “Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data,” *Proceedings of 32nd Conference on Neural Information Processing Systems*, 2018.
- [90] Liangqiong Qu, Yuyin Zhou, Paul Pu Liang, Yingda Xia, Feifei Wang, Ehsan Adeli, Li Fei-Fei, and Daniel Rubin, “Rethinking architecture design for tackling data heterogeneity in federated learning,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10061–10071.
- [91] Kangkang Wang, Rajiv Mathews, Chloé Kiddon, Hubert Eichner, Françoise Beau-fays, and Daniel Ramage, “Federated evaluation of on-device personalization,” *arXiv preprint arXiv:1910.10252*, 2019.
- [92] Tao Yu, Eugene Bagdasaryan, and Vitaly Shmatikov, “Salvaging federated learning by local adaptation,” *arXiv preprint arXiv:2002.04758*, 2020.
- [93] Filip Hanzely and Peter Richtárik, “Federated learning of a mixture of global and local models,” *arXiv preprint arXiv:2002.05516*, 2020.

- [94] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altschmidt, Sam Altman, Shyamal Anadkat, et al., “GPT-4 technical report,” *arXiv preprint arXiv:2303.08774*, 2023.
- [95] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample, “LLaMA: Open and Efficient Foundation Language Models,” 2023.
- [96] Sharan Narang and Aakanksha Chowdhery, “Pathways language model (PAML): Scaling to 540 billion parameters for breakthrough performance,” *Google AI Blog*, 2022.
- [97] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al., “Language models are few-shot learners,” *Proceedings of the 34th International Conference on Neural Information Processing System*, vol. 33, pp. 1877–1901, 2020.
- [98] Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen, “What Makes Good In-Context Examples for GPT-3?,” *Proceedings of Deep Learning Inside Out (DeeLIO 2022)*, 2021.
- [99] Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp, “Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity,” *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, 2021.
- [100] Stefan Hegselmann, Alejandro Buendia, Hunter Lang, Monica Agrawal, Xiaoyi Jiang, and David Sontag, “TabLLM: Few-shot classification of tabular data with large language models,” in *Proceedings of International Conference on Artificial Intelligence and Statistics*. PMLR, 2023, pp. 5549–5581.
- [101] Tianhao Li, Sandesh Shetty, Advait Kamath, Ajay Jaiswal, Xiaoyi Jiang, Ying Ding, and Yejin Kim, “CancerGPT for few shot drug pair synergy prediction using large pretrained language models,” *npj Digital Medicine*, vol. 7, no. 1, pp. 40, 2024.
- [102] Dongchao Yang, Haohan Guo, Yuanyuan Wang, Rongjie Huang, Xiang Li, Xu Tan, Xixin Wu, and Helen Meng, “UniAudio 1.5: Large Language Model-driven Audio Codec is A Few-shot Audio Task Learner,” *arXiv preprint arXiv:2406.10056*, 2024.

- [103] Minh-Hao Van and Xintao Wu, “Detecting and correcting hate speech in multimodal memes with large visual language model,” *arXiv preprint arXiv:2311.06737*, 2023.
- [104] Xuhai Xu, Bingsheng Yao, Yuanzhe Dong, Saadia Gabriel, Hong Yu, James Hendler, Marzyeh Ghassemi, Anind K Dey, and Dakuo Wang, “Mental-LLM: Leveraging large language models for mental health prediction via online text data,” *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 8, no. 1, pp. 1–32, 2024.
- [105] Ehsan Aghaei and Ehab Al-Shaer, “CVE-driven Attack Technique Prediction with Semantic Information Extraction and a Domain-specific Language Model,” *arXiv preprint arXiv:2309.02785*, 2023.
- [106] Xin Zhou, Ting Zhang, and David Lo, “Large language model for vulnerability detection: Emerging results and future directions,” in *Proceedings of the 2024 ACM/IEEE 44th International Conference on Software Engineering: New Ideas and Emerging Results*, 2024, pp. 47–51.
- [107] Hammond Pearce, Benjamin Tan, Baleegh Ahmad, Ramesh Karri, and Brendan Dolan-Gavitt, “Examining zero-shot vulnerability repair with large language models,” in *Proceedings of 2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2023, pp. 2339–2356.
- [108] Ruijie Meng, Martin Mirchev, Marcel Böhme, and Abhik Roychoudhury, “Large language model guided protocol fuzzing,” in *Proceedings of the 31st Annual Network and Distributed System Security Symposium (NDSS)*, 2024, vol. 2024.
- [109] Corporate News Kaspersky, “Best bite: Kaspersky reveals phishing emails that employees find most confusing,” [https://www.kaspersky.com/about/press-releases/2022\\_best-bite-kaspersky-reveals-phishing-emails-that-employees-find-most-confusing](https://www.kaspersky.com/about/press-releases/2022_best-bite-kaspersky-reveals-phishing-emails-that-employees-find-most-confusing) [Online; posted 22-June-2022].
- [110] Google Inc., “Google Safe Browsing,” <https://safebrowsing.google.com/>, [Accessed 29-07-2023].
- [111] Stephen Peters, “Microsoft Edge security for your business,” <https://learn.microsoft.com/en-us/deployedge/ms-edge-security-for-business>, [Accessed 29-07-2023].

- [112] Penghui Zhang, Adam Oest, Haehyun Cho, Zhibo Sun, RC Johnson, Brad Wardman, Shaown Sarker, Alexandros Kapravelos, Tiffany Bao, Ruoyu Wang, et al., “Crawl-phish: Large-scale analysis of client-side cloaking techniques in phishing,” in *Proceedings of 2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 1109–1124.
- [113] Maria Sameen, Kyunghyun Han, and Seong Oun Hwang, “PhishHaven—An efficient real-time AI phishing URLs detection system,” *IEEE Access*, vol. 8, pp. 83425–83443, 2020.
- [114] Routhu Srinivasa Rao, Tatti Vaishnavi, and Alwyn Roshan Pais, “PhishDump: A multi-model ensemble based technique for the detection of phishing sites in mobile devices,” *Pervasive and mobile computing*, vol. 60, pp. 101084, 2019.
- [115] Doyen Sahoo, Chenghao Liu, and Steven CH Hoi, “Malicious URL detection using machine learning: A survey,” *arXiv preprint arXiv:1701.07179*, 2017.
- [116] Pawan Prakash, Manish Kumar, Ramana Rao Kompella, and Minaxi Gupta, “Phish-net: predictive blacklisting to detect phishing attacks,” in *Proceedings of the 29th conference on Information communications*. IEEE, 2010, pp. 1–5.
- [117] Youness Mourtaji, Mohammed Bouhorma, Daniyal Alghazzawi, Ghadah Aldabbagh, and Abdullah Alghamdi, “Hybrid rule-based solution for phishing URL detection using convolutional neural network,” *Wireless Communications and Mobile Computing*, vol. 2021, pp. 1–24, 2021.
- [118] Wei Zhang, Qingshan Jiang, Lifei Chen, and Chengming Li, “Two-stage ELM for phishing Web pages detection using hybrid features,” *World Wide Web*, vol. 20, pp. 797–813, 2017.
- [119] Yan Ding, Nurbol Luktarhan, Keqin Li, and Wushour Slamun, “A keyword-based combination approach for detecting phishing webpages,” *Computers & Security*, vol. 84, pp. 256–275, 2019.
- [120] Xiuwen Liu and Jianming Fu, “SPWalk: similar property oriented feature learning for phishing detection,” *IEEE Access*, vol. 8, pp. 87031–87045, 2020.
- [121] Routhu Srinivasa Rao, Alwyn Roshan Pais, and Pritam Anand, “A heuristic technique to detect phishing websites using TWSVM classifier,” *Neural Computing and Applications*, vol. 33, pp. 5733–5752, 2021.

- [122] Guang-Gang Geng, Xiao-Dong Lee, and Yan-Ming Zhang, “Combating phishing attacks via brand identity and authorization features,” *Security and Communication Networks*, vol. 8, no. 6, pp. 888–898, 2015.
- [123] Routhu Srinivasa Rao and Alwyn Roshan Pais, “Detection of phishing websites using an efficient feature-based machine learning framework,” *Neural Computing and Applications*, vol. 31, pp. 3851–3873, 2019.
- [124] Chris McCormick, “Word2vec tutorial-the skip-gram model,” *Apr-2016.[Online]*. Available: <http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model>, 2016.
- [125] Ali Aljofey, Qingshan Jiang, Qiang Qu, Mingqing Huang, and Jean-Pierre Niyigena, “An effective phishing detection model based on character level convolutional neural network from URL,” *Electronics*, vol. 9, no. 9, pp. 1514, 2020.
- [126] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019, vol. 1, p. 2.
- [127] Garrett Wilson and Diane J Cook, “A survey of unsupervised deep domain adaptation,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 11, no. 5, pp. 1–46, 2020.
- [128] De-An Huang and Yu-Chiang Frank Wang, “Coupled dictionary and feature space learning with applications to cross-domain image synthesis and recognition,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 2496–2503.
- [129] Alan Ramponi and Barbara Plank, “Neural Unsupervised Domain Adaptation in NLP—A Survey,” *Proceedings of the 28th International Conference on Computational Linguistics*, 2020.
- [130] Cheng Wang, Mathias Niepert, and Hui Li, “RecSys-DAN: discriminative adversarial networks for cross-domain recommender systems,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 8, pp. 2731–2740, 2019.

- [131] Boqing Gong, Kristen Grauman, and Fei Sha, “Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation,” in *Proceedings of the 30th International Conference on Machine Learning*. PMLR, 2013, pp. 222–230.
- [132] Jingjing Li, Ke Lu, Zi Huang, Lei Zhu, and Heng Tao Shen, “Transfer independently together: A generalized framework for domain adaptation,” *IEEE Transactions on Cybernetics*, vol. 49, no. 6, pp. 2144–2155, 2018.
- [133] Wei Xia, Jing Huang, and John HL Hansen, “Cross-lingual text-independent speaker verification using unsupervised adversarial discriminative domain adaptation,” in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 5816–5820.
- [134] Abolfazl Farahani, Sahar Voghoei, Khaled Rasheed, and Hamid R Arabnia, “A brief review of domain adaptation,” *Advances in data science and information engineering: Proceedings from ICDATA 2020 and IKE 2020*, pp. 877–894, 2021.
- [135] Masashi Sugiyama, Matthias Krauledat, and Klaus-Robert Müller, “Covariate shift adaptation by importance weighted cross validation.,” *Journal of Machine Learning Research*, vol. 8, no. 5, 2007.
- [136] Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang, “Domain adaptation via transfer component analysis,” *IEEE Transactions on Neural Networks*, vol. 22, no. 2, pp. 199–210, 2010.
- [137] Yaniv Taigman, Adam Polyak, and Lior Wolf, “Unsupervised cross-domain image generation,” *Proceedings of International Conference on Learning Representations (ICLR)*, 2017.
- [138] Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko, “Simultaneous deep transfer across domains and tasks,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4068–4076.
- [139] Yaroslav Ganin and Victor Lempitsky, “Unsupervised domain adaptation by back-propagation,” in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*. PMLR, 2015, pp. 1180–1189.
- [140] Waqar Aqeel, Balakrishnan Chandrasekaran, Anja Feldmann, and Bruce M Maggs, “On landing and internal web pages: The strange case of jekyll and hyde in web

- performance measurement,” in *Proceedings of the ACM Internet Measurement Conference*, 2020, pp. 680–695.
- [141] Vincent Drury, Luisa Lux, and Ulrike Meyer, “Dating Phish: An analysis of the life cycles of phishing attacks and campaigns,” in *Proceedings of the 17th International Conference on Availability, Reliability and Security*, 2022, pp. 1–11.
- [142] Arthur Drichel, Vincent Drury, Justus von Brandt, and Ulrike Meyer, “Finding phish in a haystack: A pipeline for phishing classification on certificate transparency logs,” in *Proceedings of the 16th International Conference on Availability, Reliability and Security*, 2021, pp. 1–12.
- [143] 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*, 2021, pp. 1–10.
- [144] Wilfrid J Dixon and Alexander M Mood, “The statistical sign test,” *Journal of the American Statistical Association*, vol. 41, no. 236, pp. 557–566, 1946.
- [145] Leo Breiman, “Random forests,” *Machine Learning*, vol. 45, pp. 5–32, 2001.
- [146] Henry B Mann and Donald R Whitney, “On a test of whether one of two random variables is stochastically larger than the other,” *The annals of mathematical statistics*, pp. 50–60, 1947.
- [147] NAC Cressie and HJ Whitford, “How to use the two sample t-test,” *Biometrical Journal*, vol. 28, no. 2, pp. 131–148, 1986.
- [148] Morten W Fagerland and Leiv Sandvik, “The Wilcoxon–Mann–Whitney test under scrutiny,” *Statistics in medicine*, vol. 28, no. 10, pp. 1487–1497, 2009.
- [149] Google Inc., “Indexing HTTPS pages by default,” <https://developers.google.com/search/blog/2015/12/indexing-https-pages-by-default>, [Accessed 12-10-2023].
- [150] Mark A Hall, “Correlation-based feature selection of discrete and numeric class machine learning,” 2000.
- [151] OpenAI, “GPT-4 Technical Report,” 2023.

- [152] Adham Albakri, Eerke Boiten, and Rogério De Lemos, “Sharing cyber threat intelligence under the general data protection regulation,” in *Privacy Technologies and Policy: 7th Annual Privacy Forum, APF 2019, Rome, Italy, June 13–14, 2019, Proceedings 7*. Springer, 2019, pp. 28–41.
- [153] Warren B Chik, “The Singapore Personal Data Protection Act and an assessment of future trends in data privacy reform,” *Computer Law & Security Review*, vol. 29, no. 5, pp. 554–575, 2013.
- [154] Clemens Sauerwein, Christian Sillaber, Andrea Mussmann, and Ruth Breu, “Threat intelligence sharing platforms: An exploratory study of software vendors and research perspectives,” 2017.
- [155] Palo Alto Networks, “All Subscriptions - PAN-OS 11.1,” 2024, Accessed: 2025-03-22.
- [156] IBM X-Force Exchange, “IBM X-Force Exchange FAQ,” 2024, Accessed: 2025-03-22.
- [157] Richard Borden, Joshua Mooney, Mark Taylor, and Matthew Sharkey, “Threat information sharing under GDPR,” *Scitech Lawyer*, vol. 15, no. 3, pp. 30–35, 2019.
- [158] National Council of ISACs, “National Council of ISACs,” 2025, Accessed: 2025-03-28.
- [159] Wenke Huang, Mang Ye, Zekun Shi, He Li, and Bo Du, “Rethinking federated learning with domain shift: A prototype view,” in *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2023, pp. 16312–16322.
- [160] Saeed Vahidian, Sreevatsank Kadaveru, Woonjoon Baek, Weijia Wang, Vyacheslav Kungurtsev, Chen Chen, Mubarak Shah, and Bill Lin, “When do curricula work in federated learning?,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 5084–5094.
- [161] Haokun Chen, Ahmed Frikha, Denis Krompass, Jindong Gu, and Volker Tresp, “FRAug: Tackling federated learning with Non-IID features via representation augmentation,” in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023, pp. 4849–4859.
- [162] Tung-Anh Nguyen, Long Tan Le, Tuan Dung Nguyen, Wei Bao, Suranga Seneviratne, Choong Seon Hong, and Nguyen H Tran, “Federated PCA on Grassmann manifold for IoT Anomaly Detection,” *IEEE/ACM Transactions on Networking*, 2024.

- [163] Juan R Trocoso-Pastoriza, Alain Mermoud, Romain Bouyé, Francesco Marino, Jean-Philippe Bossuat, Vincent Lenders, and Jean-Pierre Hubaux, “Orchestrating collaborative cybersecurity: a secure framework for distributed privacy-preserving threat intelligence sharing,” *arXiv preprint arXiv:2209.02676*, 2022.
- [164] Talha Ongun, Simona Boboila, Alina Oprea, Tina Eliassi-Rad, Jason Hiser, and Jack Davidson, “Celest: federated learning for globally coordinated threat detection,” *arXiv preprint arXiv:2205.11459*, 2022.
- [165] Tongtong Jiang, Guowei Shen, Chun Guo, Yunhe Cui, and Bo Xie, “BFLS: Blockchain and Federated Learning for sharing threat detection models as Cyber Threat Intelligence,” *Computer Networks*, vol. 224, pp. 109604, 2023.
- [166] Davy Preuveneers, Wouter Joosen, Jorge Bernal Bernabe, and Antonio Skarmeta, “Distributed security framework for reliable threat intelligence sharing,” *Security and Communication Networks*, vol. 2020, no. 1, pp. 8833765, 2020.
- [167] Joanna Moubarak, Carole Bassil, and Joe Antoun, “On the dissemination of cyber threat intelligence through hyperledger,” in *Proceedings of 17th International Conference on the Design of Reliable Communication Networks (DRCN)*. IEEE, 2021, pp. 1–6.
- [168] Mikail Mohammed Salim, Abir El Azzaoui, Xianjun Deng, and Jong Hyuk Park, “FL-CTIF: A federated learning based CTI framework based on information fusion for secure IIoT,” *Information Fusion*, vol. 102, pp. 102074, 2024.
- [169] Chandra Thapa, Jun Wen Tang, Alsharif Abuadbba, Yansong Gao, Seyit Camtepe, Surya Nepal, Mahathir Almashor, and Yifeng Zheng, “Evaluation of federated learning in phishing email detection,” *Sensors*, vol. 23, no. 9, pp. 4346, 2023.
- [170] Ekaterina Khramtsova, Christian Hammerschmidt, Sofian Lagraa, and Radu State, “Federated learning for cyber security: SOC collaboration for malicious URL detection,” in *Proceedings of IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2020, pp. 1316–1321.
- [171] Ido Sakazi, Edita Grolman, Yuval Elovici, and Asaf Shabtai, “STFL: Utilizing a Semi-Supervised, Transfer-Learning, Federated-Learning Approach to Detect Phishing URL Attacks,” in *Proceedings of International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2024, pp. 1–10.

- [172] Yujie Li, Yanbin Wang, Haitao Xu, Zhenhao Guo, Fan Zhang, Ruitong Liu, and Wenrui Ma, “Fed-urlBERT: Client-side Lightweight Federated Transformers for URL Threat Analysis,” *arXiv preprint arXiv:2312.03636*, 2023.
- [173] Songyang Wu, Bochen Shen, Hao Zheng, Wenqi Sun, Qi Sun, and Shuang Wang, “FedURL: A BERT-based Federated Malicious URL Detection Framework,” in *Proceedings of 7th International Conference on Intelligent Computing*. Springer, 2024, pp. 122–131.
- [174] Yue Tan, Guodong Long, Jie Ma, Lu Liu, Tianyi Zhou, and Jing Jiang, “Federated learning from pre-trained models: A contrastive learning approach,” *Proceedings of 36th Conference on Neural Information Processing Systems (NeurIPS 2022)*, vol. 35, pp. 19332–19344, 2022.
- [175] Guancheng Wan, Wenke Huang, and Mang Ye, “Federated graph learning under domain shift with generalizable prototypes,” in *Proceedings of the AAAI conference on artificial intelligence*, 2024, vol. 38, pp. 15429–15437.
- [176] Yuhang Chen, Wenke Huang, and Mang Ye, “Fair federated learning under domain skew with local consistency and domain diversity,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 12077–12086.
- [177] Nikita Kotelevskii, Maxime Vono, Alain Durmus, and Eric Moulines, “FedPop: A bayesian approach for personalised federated learning,” *Proceedings of the 36th International Conference on Neural Information Processing Systems*, vol. 35, pp. 8687–8701, 2022.
- [178] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston, “Curriculum learning,” in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 41–48.
- [179] Zhuang Qi, Yuqing Wang, Zitan Chen, Ran Wang, Xiangxu Meng, and Lei Meng, “Clustering-based curriculum construction for sample-balanced federated learning,” in *Proceedings of 2nd International Conference on Artificial Intelligence*. Springer, 2022, pp. 155–166.
- [180] Amelia Jiménez-Sánchez, Mickael Tardy, Miguel A González Ballester, Diana Mateus, and Gemma Piella, “Memory-aware curriculum federated learning for breast cancer classification,” *Computer Methods and Programs in Biomedicine*, vol. 229, pp. 107318, 2023.

- [181] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, “Attention is all you need,” *Proceedings of 31st Conference on Neural Information Processing Systems (NIPS 2017)*, vol. 30, 2017.
- [182] Arvind Prasad and Shalini Chandra, “PhiUSIIL: A diverse security profile empowered phishing URL detection framework based on similarity index and incremental learning,” *Computers & Security*, vol. 136, pp. 103545, 2024.
- [183] Robert F Woolson, “Wilcoxon signed-rank test,” *Encyclopedia of biostatistics*, vol. 8, 2005.
- [184] National Coordinating Center for Communications, “National Coordinating Center for Communications,” .
- [185] Financial Services Information Sharing and Analysis Center, “FS-ISAC,” .
- [186] Kaspersky, “Kaspersky reports phishing attacks grow by 40 percent in 2023,” 2024, Accessed on June 06, 2024.
- [187] Fariza Rashid, Ben Doyle, Soyeon Caren Han, and Suranga Seneviratne, “Phishing URL detection generalisation using Unsupervised Domain Adaptation,” *Computer Networks*, vol. 245, pp. 110398, 2024.
- [188] Jessica Ryan, “COVID-19 Phishing Update: Campaigns Exploiting Hope for a Cure,” 2020, Accessed on June 06, 2024.
- [189] Francesco Greco, Giuseppe Desolda, Andrea Esposito, et al., “Explaining Phishing Attacks: An XAI Approach to Enhance User Awareness and Trust.,” in *ITASEC*, 2023.
- [190] Svetlana Bender, Samantha Horn, George Loewenstein, and Olivia Roberts, “Phishing feedback: just-in-time intervention improves online security,” *Behavioural Public Policy*, p. 1–13, 2024.
- [191] Outlook.com, “Advanced Outlook.com security for Microsoft 365 subscribers,” 2024, Accessed on September 17, 2024.
- [192] Google, “Manage warnings about unsafe sites,” 2024, Accessed on September 17, 2024.

- [193] Kholoud Althobaiti, Nicole Meng, and Kami Vaniea, “I don’t need an expert! making url phishing features human comprehensible,” in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 2021, pp. 1–17.
- [194] Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, et al., “Time-LLM: Time series forecasting by reprogramming large language models,” *Proceedings of The Twelfth International Conference on Learning Representations*, 2024.
- [195] Simran Arora, Avanika Narayan, Mayee F Chen, Laurel Orr, Neel Guha, Kush Bhatia, Ines Chami, and Christopher Re, “Ask me anything: A simple strategy for prompting language models,” in *The Eleventh International Conference on Learning Representations*, 2022.
- [196] Lefteris Loukas, Ilias Stogiannidis, Odysseas Diamantopoulos, Prodromos Malakasiotis, and Stavros Vassos, “Making LLMs worth every penny: Resource-limited text classification in banking,” in *Proceedings of the Fourth ACM International Conference on AI in Finance*, 2023, pp. 392–400.
- [197] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa, “Large language models are zero-shot reasoners,” *Proceedings of the 36th International Conference on Neural Information Processing Systems*, vol. 35, 2022.
- [198] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al., “Chain-of-thought prompting elicits reasoning in large language models,” *36th Conference on Neural Information Processing Systems (NeurIPS 2022)*, vol. 35, pp. 24824–24837, 2022.
- [199] Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu, “G-Eval: NLG evaluation using GPT-4 with better human alignment,” *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.
- [200] Sukriti Jaitly, Tanay Shah, Ashish Shugani, and Razik Singh Grewal, “Towards Better Serialization of Tabular Data for Few-shot Classification,” *arXiv preprint arXiv:2312.12464*, 2023.
- [201] Prasasthy Balasubramanian, Justin Seby, and Panos Kostakos, “Transformer-based LLMs in Cybersecurity: An in-depth Study on Log Anomaly Detection and Conversational Defense Mechanisms,” in *Proceedings of 2023 IEEE International Conference on Big Data (BigData)*. IEEE, 2023, pp. 3590–3599.

- [202] Nicholas Kroegeer, Dan Ley, Satyapriya Krishna, Chirag Agarwal, and Himabindu Lakkaraju, “Are Large Language Models Post Hoc Explainers?,” *arXiv preprint arXiv:2310.05797*, 2023.
- [203] Amrita Bhattacharjee, Raha Moraffah, Joshua Garland, and Huan Liu, “Towards LLM-guided causal explainability for black-box text classifiers,” in *Proceedings of AAAI 2024 Workshop on Responsible Language Models, Vancouver, BC, Canada*, 2024.
- [204] Yanda Chen, Ruiqi Zhong, Narutatsu Ri, Chen Zhao, He He, Jacob Steinhardt, Zhou Yu, and Kathleen McKeown, “Do models explain themselves? counterfactual simulatability of natural language explanations,” *Proceedings of the 41st International Conference on Machine Learning*, 2023.
- [205] Chirag Agarwal, Sree Harsha Tanneru, and Himabindu Lakkaraju, “Faithfulness vs. plausibility: On the (un) reliability of explanations from large language models,” *arXiv preprint arXiv:2402.04614*, 2024.
- [206] Miles Turpin, Julian Michael, Ethan Perez, and Samuel Bowman, “Language models don’t always say what they think: Unfaithful explanations in chain-of-thought prompting,” *Proceedings of the 37th International Conference on Neural Information Processing Systems*, vol. 36, 2024.
- [207] AI Anthropic, “The Claude 3 model family: Opus, Sonnet, Haiku,” *Claude-3 Model Card*, 2024.
- [208] Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al., “Gemini: A family of highly capable multimodal models,” *arXiv preprint arXiv:2312.11805*, 2023.
- [209] Meta, “Introducing Meta LLaMA 3: The most capable openly available LLM to date,” 2024, Accessed on June 08, 2024.
- [210] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin, “Why should I trust you? Explaining the predictions of any classifier,” in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016.
- [211] Yujia Zhang, Kuangyan Song, Yiming Sun, Sarah Tan, and Madeleine Udell, “Why should you trust my explanation? Understanding uncertainty in LIME explanations,”

- Proceedings of International Conference on Machine Learning AI for Social Good Workshop*, 2019.
- [212] Anil K Jain and Richard C Dubes, *Algorithms for clustering data*, Prentice-Hall, Inc., 1988.
- [213] Garima Agrawal, Kuntal Pal, Yuli Deng, Huan Liu, and Ying-Chih Chen, “CyberQ: Generating Questions and Answers for Cybersecurity Education Using Knowledge Graph-Augmented LLMs,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024, vol. 38, pp. 23164–23172.
- [214] Asli Celikyilmaz, Elizabeth Clark, and Jianfeng Gao, “Evaluation of text generation: A survey,” *arXiv preprint arXiv:2006.14799*, 2020.
- [215] Francesco Mola and Roberta Siciliano, “A fast splitting procedure for classification trees,” *Statistics and Computing*, vol. 7, pp. 209–216, 1997.
- [216] Lior Rokach and Oded Maimon, “Top-down induction of decision trees classifiers-a survey,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, vol. 35, no. 4, pp. 476–487, 2005.
- [217] Keivan Alizadeh, Iman Mirzadeh, Dmitry Belenko, Karen Khatamifard, Minsik Cho, Carlo C Del Mundo, Mohammad Rastegari, and Mehrdad Farajtabar, “LLM in a flash: Efficient large language model inference with limited memory,” *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2023.
- [218] Dominik Wagner, Alexander Churchill, Siddharth Sigtia, Panayiotis Georgiou, Matt Mirsamadi, Aarshee Mishra, and Erik Marchi, “A Multimodal Approach to Device-Directed Speech Detection with Large Language Models,” in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024, pp. 10451–10455.
- [219] Duzhen Zhang, Yahan Yu, Chenxing Li, Jiahua Dong, Dan Su, Chenhui Chu, and Dong Yu, “MM-LLMs: Recent advances in multimodal large language models,” *Findings of the Association for Computational Linguistics: ACL 2024*, 2024.
- [220] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi, “BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models,” in *Proceedings of the 40th International Conference on Machine Learning*. PMLR, 2023.

- [221] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee, “Visual instruction tuning,” *Proceedings of 37th Conference on Neural Information Processing Systems (NeurIPS 2023)*, vol. 36, 2024.
- [222] Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al., “Code LLaMA: Open foundation models for code,” *arXiv preprint arXiv:2308.12950*, 2023.
- [223] Sahar Abdelnabi, Katharina Krombholz, and Mario Fritz, “VisualPhishNet: Zero-day phishing website detection by visual similarity,” in *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, 2020, pp. 1681–1698.
- [224] Kalana Abeywardena, Jiawei Zhao, Lexi Brent, Suranga Seneviratne, and Ralph Holz, “Triplet Mining-based Phishing Webpage Detection,” in *2020 IEEE 45th Conference on Local Computer Networks (LCN)*. IEEE, 2020, pp. 377–380.
- [225] Routhu Srinivasa Rao, Amey Umarekar, and Alwyn Roshan Pais, “Application of word embedding and machine learning in detecting phishing websites,” *Telecommunication Systems*, vol. 79, no. 1, pp. 33–45, 2022.
- [226] Zhenxiang Xiao, Yuzhong Chen, Junjie Yao, Lu Zhang, Zhengliang Liu, Zihao Wu, Xiaowei Yu, Yi Pan, Lin Zhao, Chong Ma, et al., “Instruction-ViT: Multi-modal prompts for instruction learning in vision transformer,” *Information Fusion*, vol. 104, pp. 102204, 2024.
- [227] Liam Hebert, Gaurav Sahu, Yuxuan Guo, Nanda Kishore Sreenivas, Lukasz Golab, and Robin Cohen, “Multi-modal discussion transformer: Integrating text, images and graph transformers to detect hate speech on social media,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024, vol. 38, pp. 22096–22104.
- [228] Jehyun Lee, Peiyuan Lim, Bryan Hooi, and Dinil Mon Divakaran, “Multimodal large language models for phishing webpage detection and identification,” *arXiv preprint arXiv:2408.05941*, 2024.
- [229] Tri Cao, Chengyu Huang, Yuexin Li, Huilin Wang, Amy He, Nay Oo, and Bryan Hooi, “PhishAgent: a robust multimodal agent for phishing webpage detection,” *Proceedings of 39th AAAI Conference on Artificial Intelligence*, 2024.
- [230] Shangchao Su, Mingzhao Yang, Bin Li, and Xiangyang Xue, “Federated adaptive prompt tuning for multi-domain collaborative learning,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024, vol. 38, pp. 15117–15125.

- [231] Dongqi Cai, Shangguang Wang, Yaozong Wu, Felix Xiaozhu Lin, and Mengwei Xu, “Federated few-shot learning for mobile nlp,” in *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, 2023, pp. 1–17.
- [232] Liam Collins, Shanshan Wu, Sewoong Oh, and Khe Chai Sim, “Profit: Benchmarking personalization and robustness trade-off in federated prompt tuning,” *arXiv preprint arXiv:2310.04627*, 2023.

# Appendix A

## LLMs are One-Shot URL Classifiers and Explainers

In this section we first present the accuracy, recall and precision results of our one-shot classification framework using the five LLMs and three datasets as mentioned in Section 5.4. We then present a comparison of the one-shot framework and multiple supervised models trained on different combinations of the three training datasets. Finally we present the prediction performance of our proposed framework on an additional dataset, PhiUSIIL [182].

Table A.1: Prediction performance of one-shot classification - Accuracy, Precision and Recall: We observe that in terms of accuracy, precision and recall, GPT 4-Turbo returns better performance in comparison to the other LLMs.

Accuracy						
Test Dataset	URLTran (Supervised)	GPT 4	Claude 3	Gemini	LLaMA 3	LLaMA 2
HP	0.99	<b>0.93</b> ± 0.00174	0.87 ± 0.00089	0.85 ± 0.00666	0.81 ± 0.00390	0.47 ± 0.00615
EBBU	0.99	<b>0.95</b> ± 0.00338	0.90 ± 0.00223	0.83 ± 0.01288	0.85 ± 0.00280	0.49 ± 0.00906
ISCX	0.99	<b>0.87</b> ± 0.00458	0.83 ± 0.00388	0.73 ± 0.00408	0.76 ± 0.00589	0.44 ± 0.00970
Precision						
Test Dataset	URLTran (Supervised)	GPT 4	Claude 3	Gemini	LLaMA 3	LLaMA 2
HP	0.99	<b>0.94</b> ± 0.00204	0.90 ± 0.00048	0.89 ± 0.00772	0.87 ± 0.00298	0.72 ± 0.03389
EBBU	0.99	<b>0.95</b> ± 0.00397	0.92 ± 0.00153	0.86 ± 0.01407	0.89 ± 0.00227	0.76 ± 0.01143
ISCX	0.99	<b>0.87</b> ± 0.00530	0.87 ± 0.00275	0.80 ± 0.01118	0.84 ± 0.00408	0.62 ± 0.13112
Recall						
Test Dataset	URLTran (Supervised)	GPT 4	Claude 3	Gemini	LLaMA 3	LLaMA 2
HP	0.99	<b>0.93</b> ± 0.00174	0.87 ± 0.00089	0.85 ± 0.00666	0.81 ± 0.00390	0.47 ± 0.00666
EBBU	0.99	<b>0.95</b> ± 0.00338	0.90 ± 0.00223	0.83 ± 0.01288	0.85 ± 0.00280	0.49 ± 0.01288
ISCX	0.99	<b>0.87</b> ± 0.00458	0.83 ± 0.00388	0.73 ± 0.00408	0.76 ± 0.00589	0.44 ± 0.00408

Table A.2: Comparing performance of LLM one-shot classification with supervised models trained on combinations of training data. Here Simple Supervised refers to the supervised model URLTran train on the training datasets corresponding to the test sets. The One-shot model refers to our proposed one-shot classification framework using GPT 4-Turbo and the rest of the models are supervised URLTran on combinations of the training datasets. We observe that among the supervised models, the models trained on combinations of the training data performs poorly in comparison to the Simple Supervised model. This reflects the effect of the difference in distributions between the training and test sets. In general we also observe that the one-shot performance is comparable to all variations of supervised models. For example in terms of F1 score, for the HP and EBBU test sets, the one-shot performance is better than the supervised performance when the test set is not represented in the training set.

F1 Score						
Test Data	Simple Supervised	HP-EBBU	EBBU-ISCX	HP-ISCX	HP-EBBU-ISCX	One-shot
HP	0.99	0.97	0.91	0.94	0.94	0.94
EBBU	0.99	0.95	0.92	0.67	0.92	0.95
ISCX	0.99	0.84	0.97	0.86	0.92	0.87
Precision						
Test Data	Simple Supervised	HP-EBBU	EBBU-ISCX	HP-ISCX	HP-EBBU-ISCX	One-shot
HP	0.99	0.96	0.96	0.99	0.93	0.94
EBBU	0.99	0.95	0.96	0.95	0.88	0.95
ISCX	0.99	0.78	1.00	1.00	0.95	0.87
Recall						
Test Data	Simple Supervised	HP-EBBU	EBBU-ISCX	HP-ISCX	HP-EBBU-ISCX	One-shot
HP	0.99	0.98	0.87	0.89	0.95	0.93
EBBU	0.99	0.96	0.89	0.52	0.97	0.95
ISCX	0.99	0.90	0.95	0.75	0.90	0.87
Accuracy						
Test Data	Simple Supervised	HP-EBBU	EBBU-ISCX	HP-ISCX	HP-EBBU-ISCX	One-shot
HP	0.99	0.97	0.92	0.94	0.94	0.93
EBBU	0.99	0.95	0.93	0.75	0.92	0.95
ISCX	0.99	0.83	0.98	0.88	0.93	0.87

Table A.3: Prediction performance of one-shot classification for PhiUSIIL [182]: To evaluate our methodology in Section 5.4 we used the three datasets HP, EBBU and ISCX. Our selection of EBBU and ISCX was based on their high number of citations while we included the HP dataset introduced in our previous work [187] due to its recency. To further our evaluation we ran our experiments on an additional dataset: PhiUSIIL [182]. This dataset consists of legitimate URLs collected from Open PageRank Initiative and phishing URLs collected from PhishTank, OpenPhish, and MalwareWorld, and therefore represents a different domain of phishing and benign URLs in comparison to HP, EBBU and ISCX. As it is a more recent dataset with fewer citations we include the results in the Appendix and present the results above.

Metric	Models					
	URLTran (Supervised)	GPT 4	Claude 3	Gemini	LLaMA 3	LLaMA 2
Accuracy	0.99	0.87	0.81	0.72	0.71	0.50
Precision	0.99	0.85	0.93	0.86	0.93	0.30
Recall	0.99	0.91	0.66	0.52	0.46	0.0024
F1	0.99	0.88	0.77	0.65	0.61	0.0048