National College of Ireland

An Application of Explainable AI on Dynamic Convolutional Neural Networks for Transparent Malicious URL Detection

MSc Research Project
MSc Cybersecurity

Maame Yaa Osei
Student ID: x21176183

School of Computing
National College of Ireland

Supervisor: Ross Spellman

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Maame Yaa Osei
Student ID:	x21176183
Programme:	MSc Cybersecurity
Year:	2024
Module:	MSc Research Project
Supervisor:	Ross Spellman
Submission Due Date:	14/12/2023
Project Title:	An Application of Explainable AI on Dynamic Convolutional Neural Networks for Transparent Malicious URL Detection
Word Count:	4934
Page Count:	20

Signature:	
Date:	30th January 2024

An Application of Explainable AI on Dynamic Convolutional Neural Networks for Transparent Malicious URL Detection

Maame Yaa Osei
x21176183

Abstract

Artificial Intelligence (AI) and machine learning models, particularly in cybersecurity, are experiencing rapid growth and widespread adoption. This surge, often involving expanding parameters to improve efficiency, gives rise to a deeper understanding of such complex models. The rapid advancement of Artificial Intelligence (AI) and machine learning in cybersecurity births the need for a deeper understanding of these technologies. This research focuses on Dynamic Convolutional Neural Networks (DCNNs), using Explainable AI (XAI) to clarify their complexities. DCNNs, known for their exceptional feature extraction abilities, dynamically adjust during convolution to detect intricate data patterns.

Understanding the complex inner workings of DCNNs is challenging. The research introduces tools like Local Interpretable Model-agnostic Explanations (LIME) within XAI to provide insights into the models' decision processes, including feature importance. This approach also integrates word and character embedding to capture linguistic and morphological subtleties.

Incorporating XAI into DCNN models transforms cybersecurity practices, enabling security professionals to make more informed decisions in building and refining machine learning models and enhancing threat detection and response. As machine learning models increase in complexity, the significance of XAI in ensuring their reliability, transparency, and effectiveness becomes paramount. This study emphasizes XAI's critical role in making cybersecurity AI models more understandable, efficient, and transparent, contributing to better handling of evolving cybersecurity threats.

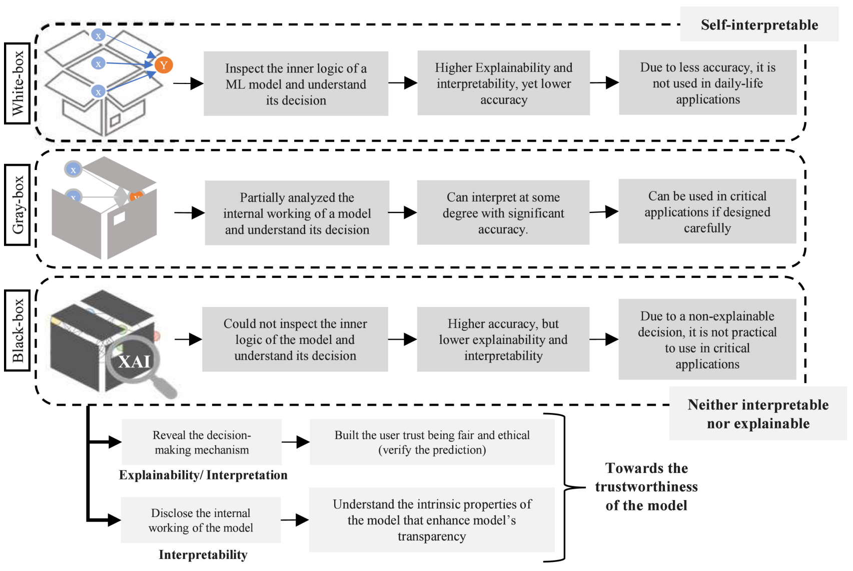
1 Introduction

In the ever-evolving digital landscape, malicious URLs pose a significant threat to individuals, organizations, and society Abad et al. (2023); Wang et al. (2021); Saleem Raja et al. (2021). These malicious links can lead to phishing attacks, malware infections, and data breaches, causing substantial financial losses and reputational damage Jones (2021). Effective and reliable URL detection methods are crucial to combat this growing threat.

Traditional URL detection methods, such as keyword-based filtering and blacklisting, have proven to be insufficient as attackers become increasingly sophisticated in their techniques. These methods often rely on predefined rules and signatures, making them vulnerable to evasion by attackers who rapidly employ new obfuscation techniques.

Dynamic Convolutional Neural Networks (DCNNs) have emerged as a promising approach for URL detection, offering superior performance and adaptability compared to traditional methods. DCNNs can effectively extract complex patterns from URL data, enabling them to identify malicious URLs more accurately.

However, despite their effectiveness, DCNNs are often considered black boxes, making it challenging to understand their decision-making processes. This lack of explainability can hinder trust and adoption in these models, as assessing their reliability and identifying potential biases becomes difficult. The figure below captures the relationship between the white-box model, grey-box, and black-box, showcasing their explainability alongside their performance. The figure depicts a high performance in terms of accuracy in black-box models but minimal interpretations.



Self-interpretable
White-box
Inspect the inner logic of a ML model and understand its decision
Higher Explainability and interpretability, yet lower accuracy
Due to less accuracy, it is not used in daily-life applications
Gray-box
Partially analyzed the internal working of a model and understand its decision
Can interpret at some degree with significant accuracy.
Can be used in critical applications if designed carefully
Black-box
XAI
Could not inspect the inner logic of the model and understand its decision
Higher accuracy, but lower explainability and interpretability
Due to a non-explainable decision, it is not practical to use in critical applications
Neither interpretable nor explainable
Reveal the decision-making mechanism
Explainability/ Interpretation
Built the user trust being fair and ethical (verify the prediction)
Disclose the internal working of the model
Interpretability
Understand the intrinsic properties of the model that enhance model's transparency
Towards the trustworthiness of the model

Figure 1: Comparison of white-box, grey-box and black-box models

Explainable AI (XAI) Ali et al. (2023) tools offer a solution to this challenge by providing insights into the inner workings of complex models like DCNNs. These tools can help us understand the model's predictions, identify essential features, and detect potential biases. By incorporating XAI tools, we can enhance the transparency and trustworthiness of DCNN-based URL detection systems. Especially in the domain of cybersecurity, where analysis and reporting are crucial for auditing and regulatory compliance, XAI tools allow cybersecurity professionals to understand the inner workings of models, to meet the requirements of the dynamic cyber landscape Keita (2023).

1.1 Background and Context of Malicious URL Detection

The internet's rapid growth and the increasing number of online services have created ample avenues for cybercriminals to exploit. Malicious URLs serve as a standard entry point for cyberattacks. They often masquerade as legitimate links to deceive unsuspecting users Jang-Jaccard and Nepal (2014). Phishing attacks, a prevalent form of cybercrime, utilize malicious URLs to lure victims into providing sensitive information, such as login credentials or financial details Rameem Zahra et al. (2022); Li et al. (2019); Lemay et al. (2020). These attacks often involve sending emails or displaying fake web pages that closely resemble genuine ones, prompting users to enter their personal information. Malware infections, another significant threat, can be triggered by clicking on malicious URLs. These URLs can lead to the download of malicious software, which can compromise user devices and networks, and steal sensitive data. The consequences of malicious URL attacks can be severe, ranging from financial losses and data breaches to reputational damage and operational disruptions. Protecting against these threats is paramount for the safety of individuals, organizations, and society.

1.2 Research Objectives and Questions

This research aims to investigate the application of XAI tools in enhancing the transparency and explainability of DCNN-based URL detection systems. The primary objectives are to:

1. Develop a DCNN-based URL detection model with high accuracy and robustness.
2. Integrate the XAI tool, such as LIME (Local Interpretable Model-Agnostic Explanations), into the DCNN model to provide insights into its decision-making processes.
3. Analyze the impact of XAI tools on the interpretability and trustworthiness of the DCNN model.
4. Evaluate the effectiveness of the XAI-enhanced DCNN model in detecting malicious URLs in real-world scenarios.

2 Related Work

2.1 Malicious URL Detection

Over the years, extensive research has gone into efforts to address the escalating cybersecurity threats Jang-Jaccard and Nepal (2014); Patgiri et al. (2023); Mondal et al. (2021) posed by malicious URLs. However, more sophisticated techniques are being employed by threat actors to gain access to systems. Early attempts to combat malicious URLs mainly relied on blacklisting techniques Sun and Liu (2023), which included maintaining a list of known malicious URLs for blocking access. While blocking offers simplicity, it is vulnerable to evasion through techniques like domain shadowing and URL obfuscation.

Blacklisting is further complicated by the use of tools like domain generation algorithms (DGAs)Sun and Liu (2023), where malicious actors employ dynamic generation of domain names to evade traditional detection methods, adding complexity to the process of identifying malicious URLs.

Prior research Palmer (2019), shows a close correlation between malicious links and cyber intrusions. The most notable types of malicious URLs included malware links, phishing, and deformed URLs.

Abad et al. (2023) contributed to a better understanding of their characteristics and patterns. Phishing URLs often exhibit deceptive elements mimicking legitimate websites, malware URLs may display patterns associated with known malicious activities, deformed URLs pose challenges with altered structures, and studying benign URLs establishes a baseline for normal behaviour, aiding in anomaly detection.

In response to the shortcomings of blacklisting, researchers turned to machine learning (ML) techniques to enhance malicious URL detection. With their ability to analyze vast datasets and identify patterns, ML algorithms demonstrated increased accuracy and adaptability in classifying URLs as benign or malicious.

Support Vector Machines (SVMs)Naresh et al. (2020), recognized for their optimal hyperplane establishment for separating data points, proved effective in malicious URL detection. The robustness and high-dimensional data handling capabilities of Random Forests (RFs), Alsaedi et al. (2022) an ensemble learning algorithm, also gained prominence in URL classification tasks. Deep Neural Networks (DNNs), particularly Convolutional Neural Networks (CNNs), emerged as powerful tools, leveraging dynamic feature extraction to identify complex patterns associated with malicious URLs.

While ML models proved effective, their black-box nature raised interpretability and trust concerns. Researchers have indulged in trying to explain these models, leading to Explainable AI (XAI) techniques being introduced to shed light on the decision-making processes of ML models. XAI tools, applied to feature extraction models like Support Vector Machines and Random Forests, reveal specific features influencing a URL's classification, aiding in identifying biases or limitations in the feature extraction process. Adapting XAI techniques to CNNs, traditionally applied to image-related studies, offers insights into the dynamic extraction of features from textual data, providing a crucial understanding of how CNNs process information for malicious URL detection.

Recent advancements in malicious URL detection focus on overcoming challenges and limitations, including real-time detection systems, addressing class imbalances Chuang et al. (2023) in datasets, integrating contextual information for enhanced accuracy, and exploring deep learning models like CNNs for improved feature extraction. XAI techniques play a pivotal role in improving transparency and trust in the decision-making processes of ML models, especially in the context of CNNs applied to textual data for malicious URL detection.

In summary, the convergence of CNNs and XAI represents a significant leap forward in understanding deep learning models' dynamic feature extraction processes to identify malicious URLs. This research provides critical insights into the intricate decision-making mechanisms of CNNs, traditionally explored in image-related studies when applied to the nuanced realm of malicious URL detection.

2.2 Explainable AI (XAI) and Its Applications

Explainable AI (XAI) Chuang et al. (2023) refers to a set of techniques and methodologies that aim to make the decision-making processes of AI models more transparent and understandable. XAI enhances trust, identifies potential biases, and facilitates debugging by providing insights into how these models make predictions. XAI techniques can be broadly categorized into two main types: model-specific and model-agnostic Meske et al. (2022). Model-specific methods are tailored to a particular model architecture, while model-agnostic methods can be applied to any model, regardless of its structure. XAI techniques have proven useful in demystifying the intricacies of a wide range of machine-learning solutions. Various XAI techniques have been applied in the use case of malicious URL detection. However, there still remains an opportunity to apply them to DCNN-based malicious URL detection schemes, and the assessment of their performances.

2.3 DCNNs in Cybersecurity

Dynamic Convolutional Neural Networks (DCNNs) Sengupta (2021); Wejinya and Bhatia (2021) have emerged as a powerful approach for URL detection, offering improved performance and adaptability compared to traditional methods. DCNNs can effectively capture complex patterns and extract meaningful features from URL data, enabling them to identify malicious URLs more accurately. DCNNs have been successfully applied in various cybersecurity applications, including malware detection, network intrusion detection, and phishing attack identification. Their ability to learn from large amounts of data and adapt to evolving threats makes them well-suited for URL detection challenges. The use of XAI in DCNN-based URL detection is paramount for several reasons:

1. **Transparency and Trust:** XAI provides insights into the decision-making processes of DCNNs, enhancing transparency and trust in these models. This is especially important in security-critical applications where understanding how a model makes predictions is crucial

2. **Bias Detection:** XAI can help identify potential biases in DCNN models, ensuring they are fair and unbiased in their decision-making. This is critical for avoiding discrimination and unfair outcomes.

3. **Debugging and Improvement:** XAI can facilitate debugging and improvement of DCNN models by highlighting essential features and identifying areas for improvement. This can lead to more accurate and robust models.

4. **Reporting and Analysis:** XAI provides valuable insights for reporting and analysis, enabling cybersecurity professionals to understand better the nature of malicious URLs and the factors contributing to misclassifications.

3 Methodology

The research employed a dataset comprising 651,191 URLs categorized into four classes: 428,103 benign URLs, 96,457 defacement URLs, 94,111 phishing URLs, and 32,520 malware URLs, sourced from Kaggle [1]. The Dynamic Convolutional Neural Network (DCNN) incorporates advanced techniques to effectively process URL data. In the initial step of tokenization, URLs are segmented into word and character sequences. Tokenization is the process of breaking down a sequence of text into individual units, such as words or characters, to facilitate further analysis. This allows the model to understand and process the structural components of URLs. Following tokenization, the model utilizes word and character embeddings to create dense vector representations of the URL features. Embeddings are numerical representations of words or characters in a continuous vector space. Word embeddings capture semantic relationships between words, representing similar meanings with similar vectors. This is valuable for comprehending the context and meaning of words in a given URL. Character embeddings, on the other hand, capture the structural information at a finer level. This is crucial when dealing with variations such as misspellings at the character level. By combining both word and character embeddings, the model gains the ability to learn a more comprehensive and robust representation of

[1]Dataset from kaggle.com: `https://www.kaggle.com/code/habibmrad1983/habib-mrad-detection-malicious-url-using-ml-models`

the input data, enabling it to discern patterns and variations in URLs effectively. The inclusion of dynamic convolutional layers in the architecture is pivotal for local pattern capturing. Unlike traditional convolutional layers with fixed filter sizes, dynamic convolutional layers adaptively adjust their filter sizes based on the input sequence. This adaptability enables the model to learn and recognize patterns of different sizes, enhancing its capability to capture intricate details within the sequential data of URLs. The model further employs max pooling as a feature extraction technique. Max pooling involves selecting the maximum value within a window, contributing to downsampling, and retaining the most crucial information from the output of convolutional layers. This process aids in focusing the model's attention on the most relevant features within the URL sequences Chaudhuri (2021). For final feature extraction and integration, the architecture incorporates fully connected dense layers. These layers densely establish connections between neurons, allowing the model to learn complex relationships and representations of the features extracted by previous layers. This step plays a crucial role in consolidating the learned information and making the final decisions based on the processed URL data. During validation, the dataset is split, and the model's performance is evaluated to ensure it generalizes well to unseen data. To refine the model, an optimization process is employed with a specified number of epochs set at five, along with a batch size of 64 for efficient processing. This iterative refinement ensures that the model's parameters are adjusted to improve its ability to recognize and classify patterns in URL data effectively.

3.1 LIME for Explainability

To enhance the interpretability of the DCNN model, an XAI tool, LIME (Local Interpretable Model-Agnostic Explanations), was integrated. These tools provide insights into the model's decision-making process, allowing for a better understanding of the factors influencing its predictions. LIME operates on individual URL instances, explaining why the model classifies a URL as either benign, defacement, phishing, or malware. They are able to achieve this by generating local approximation models that mimic the behavior of the original DCNN model for specific URLs. The explanations provided by LIME highlight the relative importance of different features, such as specific words, characters, or URL patterns, in influencing the model's prediction for that particular URL Ibrahim and Shafiq (2023).

4 Design Specification

The URL detection system implemented by this paper aims to accurately classify URLs, such as benign, deformed, phishing, and malware, based on their characteristics and potential malicious intent. The system utilizes a Deep Convolutional Neural Network(DCNN) architecture to extract meaningful features from URLs and classify them into categories. To enhance the interpretability of the DCNN's decision-making process, XAI techniques are employed to provide explanations for each classification.

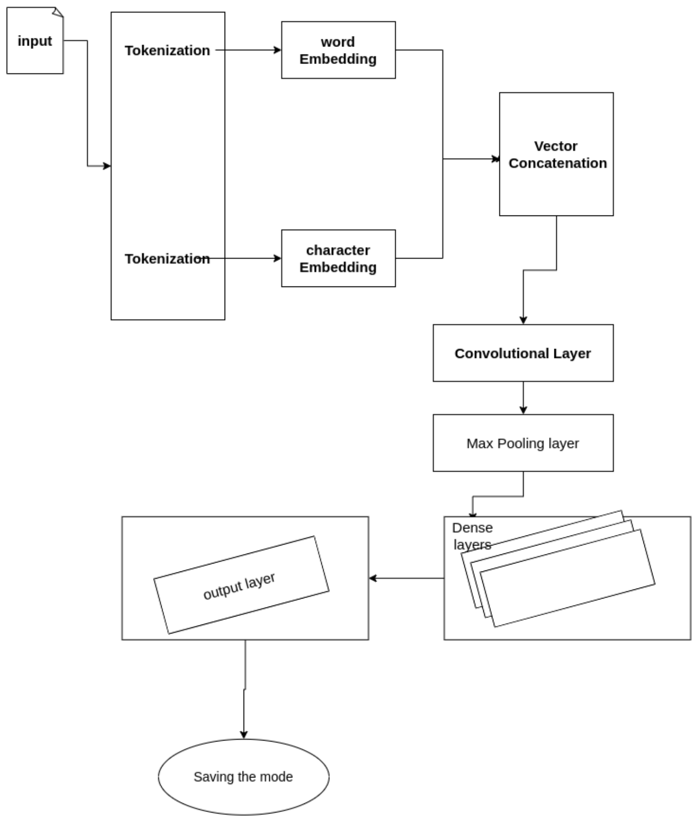


Figure 2: Architecture of a Dynamic Convolutional Neural Network

5 Implementation

The implementation of the work of this paper involved transforming raw URL data into a format suitable for the Dynamic Convolutional Neural Network (DCNN) model, incorporating various steps. Firstly, the dataset was loaded using the Pandas library, obtaining the URL texts and their corresponding labels. This step laid the foundation for subsequent data transformations. Next, in the preprocessing phase, tokenization was implemented to split the URLs into individual words and characters, creating sequences for both word-level and character-level analyses. Tokenization not only facilitated the conversion of text to lower case and removal of punctuation but also handled the elimination of words occurring less than twice, those occurring in more than 50 per cent of the texts, and stop words. These measures ensured a clean and standardized representation of the data.

Following tokenization, embedding techniques were applied using the Keras Tokenizer. This involved the creation of a word index based on word frequency in the dataset. Each word in the text was then converted into its corresponding integer index. Pad sequences were employed to ensure uniform sequence lengths, with sequences shorter than the specified length padded with zeros. Additionally, a label dictionary was created to map each label to an integer, facilitating the conversion of labels into one-hot encodings which allow the various classification categories to be easily interpreted by the model. The resulting one-hot encodings were crucial for training the DCNN model, as input texts and labels are expected to be in numerical form.

Subsequently, the training and model development phase was initiated using the TensorFlow framework. The DCNN model architecture comprised various layers, including embedding, dynamic convolutional, max-pooling, and fully connected dense layers. The training process, powered by the Adam optimizer, underwent hyperparameter tuning to optimize the model's performance.

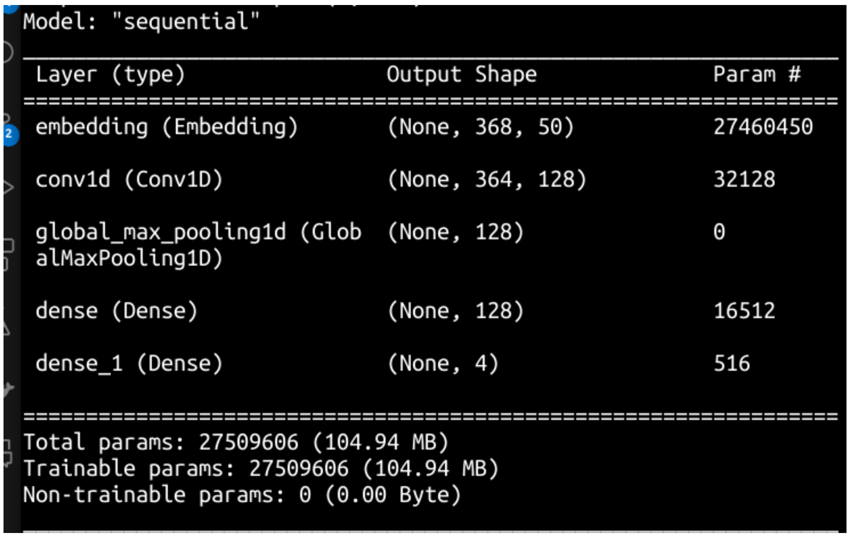
Finally, to enhance the interpretability of the DCNN model, Local Interpretable Model-agnostic Explanations (LIME) was incorporated. These Explainable Artificial Intelligence (XAI) tools provided invaluable insights into the model's decision-making process, clearly understanding the factors influencing its predictions. LIME and generated local approximation models that mirrored the behaviour of the original DCNN model for specific instances. The final outputs of the implementation included a trained DCNN model ready for real-time URL classification, as well as LIME and explanations for individual URL predictions. These outputs were achieved using tools and resources, including Python, TensorFlow, LIME and a dataset of 651,191 URLs obtained from Kaggle. The amalgamation of these tools and resources ensured the successful realization of the work of this paper.

6 Evaluation

This section elaborates on the implementations of the paper, starting with a naive approach and detailing some of the most notable changes in the design. Experiments were conducted involving minor adjustments, such as adjusting the epochs, batch size, and more. However, significant changes in experiments, minor adjustments, and findings for each section will be addressed in the experiments.

6.1 Experiment 1

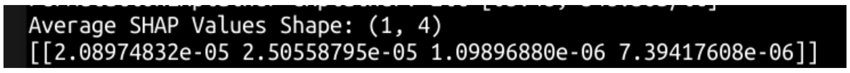
In the initial experiment conducted, a simple DCNN model capable of classifying URLs into distinct categories: benign, deformed, phishing, or malware was developed. Below is a simple summary of the model



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 368, 50)           27460450

 conv1d (Conv1D)             (None, 364, 128)          32128

 global_max_pooling1d (Glob  (None, 128)               0
 alMaxPooling1D)

 dense (Dense)               (None, 128)               16512

 dense_1 (Dense)             (None, 4)                 516

=================================================================
Total params: 27509606 (104.94 MB)
Trainable params: 27509606 (104.94 MB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 3: A Summary Of The Simple DCNN

The simplicity of the model's architecture demonstrated its effectiveness, achieving an impressive accuracy rate of 95%. Following a Sequential structure, a common approach in neural network design, the model consisted of critical layers. The Embedding Layer was crucial in transforming input URL sequences into dense vector representations, capturing semantic meaning and morphological information. The Convolutional Layer employed filters to extract local patterns and dependencies within URL sequences, introducing non-linearity with the 'relu' activation function. The Global Max Pooling Layer reduced feature map dimensionality, preserving essential information. The Dense Layer performed final feature processing, aligning dimensions with the number of output classes, and the Output Layer applied a softmax activation function, determining the predicted class based on the highest probability. To demonstrate how the model predicted, amongst the four classes, below is a sample prediction output



```
Average SHAP Values Shape: (1, 4)
[[2.08974832e-05 2.50558795e-05 1.09896880e-06 7.39417608e-06]]
```

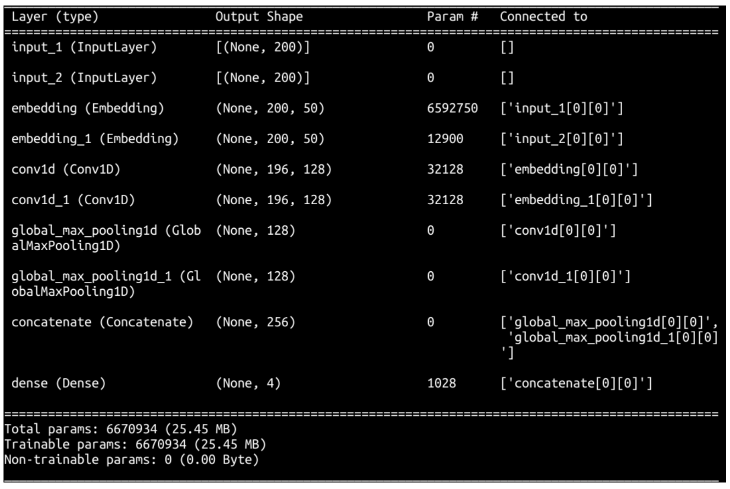
Figure 4: List Of Class Predictions

From the above, the model would take the highest value as the prediction classification. In this case, the model predicts with a 73 per cent probability that the URL is defacement based on the input list i.e ['malware', 'phishing', 'benign', 'defacement']

However, the experiment revealed challenges in feature extraction due to the diverse and unstructured nature of URL data. URLs encompass a mixture of words, characters, and symbols with irregular patterns, posing difficulties in capturing nuanced relationships. While the naive model demonstrated deep learning's potential in URL detection, achieving a 95% accuracy, it underscored the need for more sophisticated feature extraction techniques. Further experimentation and model refinement were crucial to address these challenges and enhance classification accuracy. Experiment 1 laid the groundwork for deep learning in URL detection, emphasizing the importance of advanced feature extraction to elevate model performance in real-world scenarios.

6.2 Experiment 2

This experiment delved into a more intricate architecture by integrating word and character embedding to refine our URL detection model. This advanced model demonstrated notable success, boasting an accuracy rate of 96.7%. Incorporating richer linguistic information showcased the potential for leveraging diverse linguistic features in URL detection. Below is the model's summary:



```
 Layer (type)                Output Shape                 Param #   Connected to
==================================================================================================
 input_1 (InputLayer)        [(None, 200)]                0         []

 input_2 (InputLayer)        [(None, 200)]                0         []

 embedding (Embedding)       (None, 200, 50)              6592750   ['input_1[0][0]']

 embedding_1 (Embedding)     (None, 200, 50)              12900     ['input_2[0][0]']

 conv1d (Conv1D)             (None, 196, 128)             32128     ['embedding[0][0]']

 conv1d_1 (Conv1D)           (None, 196, 128)             32128     ['embedding_1[0][0]']

 global_max_pooling1d (Glob  (None, 128)                  0         ['conv1d[0][0]']
 alMaxPooling1D)

 global_max_pooling1d_1 (Gl  (None, 128)                  0         ['conv1d_1[0][0]']
 obalMaxPooling1D)

 concatenate (Concatenate)   (None, 256)                  0         ['global_max_pooling1d[0][0]',
                                                                     'global_max_pooling1d_1[0][0]
                                                                    ']

 dense (Dense)               (None, 4)                    1028      ['concatenate[0][0]']

==================================================================================================
Total params: 6670934 (25.45 MB)
Trainable params: 6670934 (25.45 MB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 5: A Summary Of The Complex DCNN Architecture

The model's architecture embraced a multi-input design, enabling independent processing of word and character sequences before merging their feature representations. Critical components of this architecture included the Word Embedding Layer, which transformed words into dense vector representations, capturing semantic meaning. The Character Embedding Layer was also introduced to embed individual characters, addressing morphological intricacies and accommodating misspellings or typos. Distinct convolutional layers were applied to the outputs of word and character embeddings, extracting local patterns and dependencies within each sequence. Global Max Pooling Layers followed, reducing feature map dimensionality while retaining salient information. The outputs from word and character pooling were merged, creating a comprehensive feature representation that integrated both word-level and character-level insights. Subsequent processing involved a dense layer for refinement and an output layer applying a softmax activation function to produce a probability distribution across the four classes. Experiment 2 presented significant improvements over the naive model. By incorporating word and character embedding, the model captured a more nuanced representation of URL features, encompassing semantics and morphology. Enhanced feature extraction was achieved through separate convolutional layers, enabling the model to discern patterns at granular and linguistic levels, ultimately improving accuracy. However, challenges persisted, including efficient data extraction from large datasets, increased model complexity with the multi-input architecture, and the need for careful hyperparameter tuning. Below is a demonstration of the features the model extracted, which proved very inefficient in explaining its classifications:

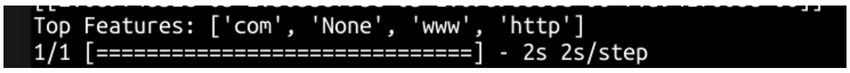

```
Top Features: ['com', 'None', 'www', 'http']
1/1 [==============================] - 2s 2s/step
```

Figure 6: Features Extracted Without An Intermediate Layer

Despite these challenges, the combined model demonstrated its effectiveness in URL detection, underscoring the importance of extracting diverse linguistic information to elevate overall model performance. Experiment 2 marked a pivotal step forward, emphasizing the advantages of integrating word and character embedding for enhanced URL detection accuracy.

6.3 Experiment 3

In the evolution of the model's architecture, Experiment 3 built upon the successes of Experiment 2 by introducing an intermediate layer. This additional layer, strategically positioned between the convolutional and dense layers, was pivotal in enabling the explainable AI (XAI) techniques proposed for usage in this paper, LIME. This was useful in facilitating the extraction of more meaningful features, as shown in the image below:

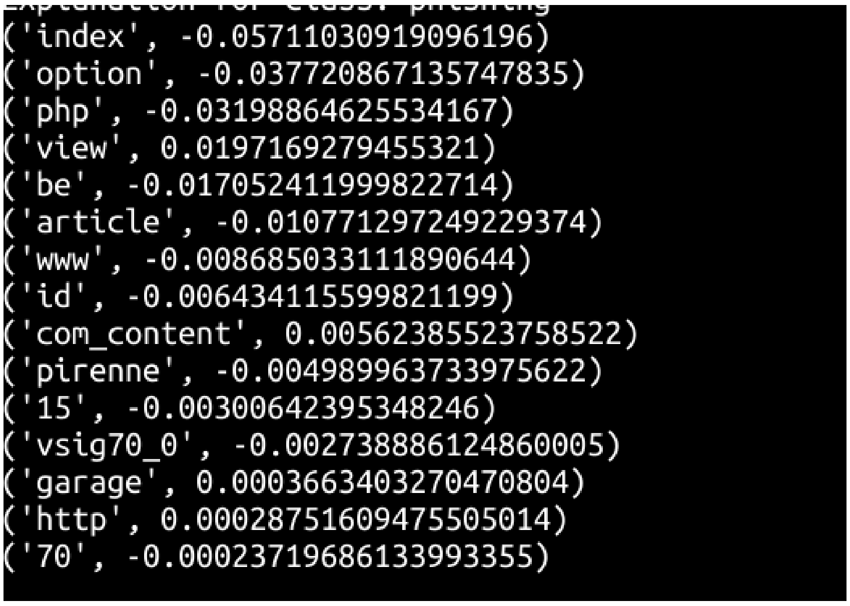

```
Explanation for class: phishing
('index', -0.05711030919096196)
('option', -0.037720867135747835)
('php', -0.03198864625534167)
('view', 0.0197169279455321)
('be', -0.017052411999822714)
('article', -0.010771297249229374)
('www', -0.008685033111890644)
('id', -0.006434115599821199)
('com_content', 0.00562385523758522)
('pirenne', -0.004989963733975622)
('15', -0.00300642395348246)
('vsig70_0', -0.002738886124860005)
('garage', 0.0003663403270470804)
('http', 0.0002875160947550 5014)
('70', -0.0002371968613399 3355)
```

Figure 7: Features Extracted With Intermediate Layer

The motivation behind incorporating the intermediate layer stemmed from a crucial limitation observed in previous models—their inability to extract meaningful features for LIME explanations. These XAI techniques heavily rely on understanding the contributions of individual input features to the model's predictions. However, the sequential architecture of the initial models posed challenges in isolating these contributions. The intermediate layer served as a crucial checkpoint within the model, capturing the output of the convolutional layers before the final dense layer transformation. This intermediate output encapsulated the most salient features extracted from the input URL sequences, laying the foundation for LIME to delve into and analyze the intricate decision-making process of the model. LIME, leveraging the intermediate output, provided explanations for the model's predictions on individual URLs. Applying their respective algorithms to this intermediate output, these XAI techniques attributed the model's predictions to the contributions of unique input features. However, Experiment 3 encountered some challenges in feature interpretation due to the skewness of the data. As a result, experiment 3 required a reduction of the dataset to 10,000 URLs: 7324 benign, 1809 defacement, 610 phishing, and 256 malware.

Below is a graphical representation of the above.

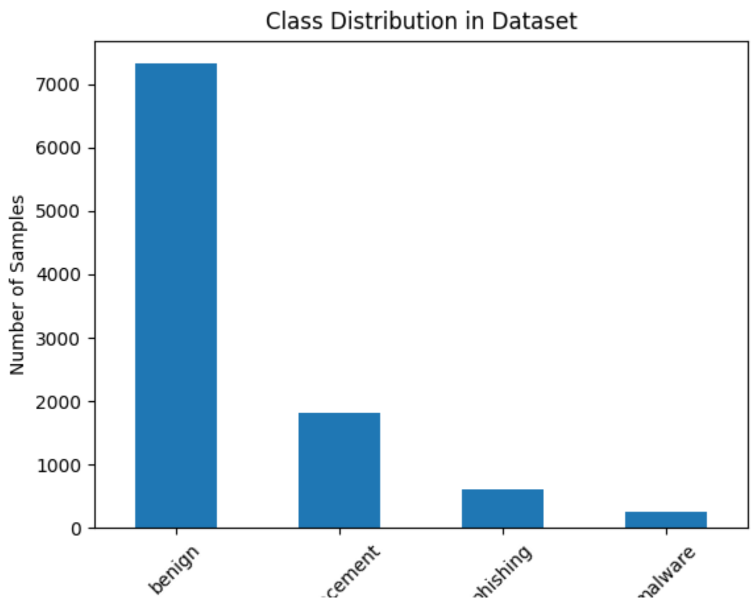


Figure 8: Unbalanced Dataset Distribution

Using the URL "espn.go.com/nba/player/-/id/3457/brandon-rush," the model trained with the unbalanced dataset classified it as 'benign', which is correct. However, the LIME explanation for the URL produced explanations for the "phishing" class, which can be attributed to the inner workings of the LIME tool that retrieved explanations for the most available classification. To test the biases of this model, I used ten random URLs, and from the sample dataset, I noted that out of 10 tests, the lime explanations provided seven explanations for "phishing" and the other three for "benign".

Below is an image showing how the URL "espn.go.com/nba/player/-/id/3457/brandon-rush," was interpreted

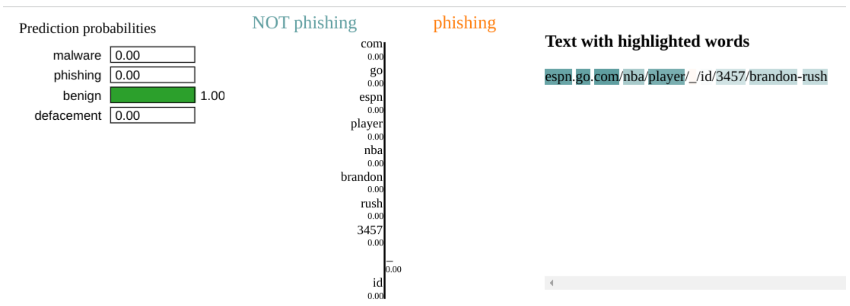


Figure 9: Lime Explanation of The Unbalanced Dataset

From the evaluation above, it is clear that the model classified the URL as "benign" however, it produced lime explanation for the class "phishing". This can be attributed to the ability of XAI tools to gravitate towards the dominant classes in the data, reflecting the model's inherent bias towards certain classifications.

Below is a more detailed demonstration of LIME explanations on feature influence on the phishing category

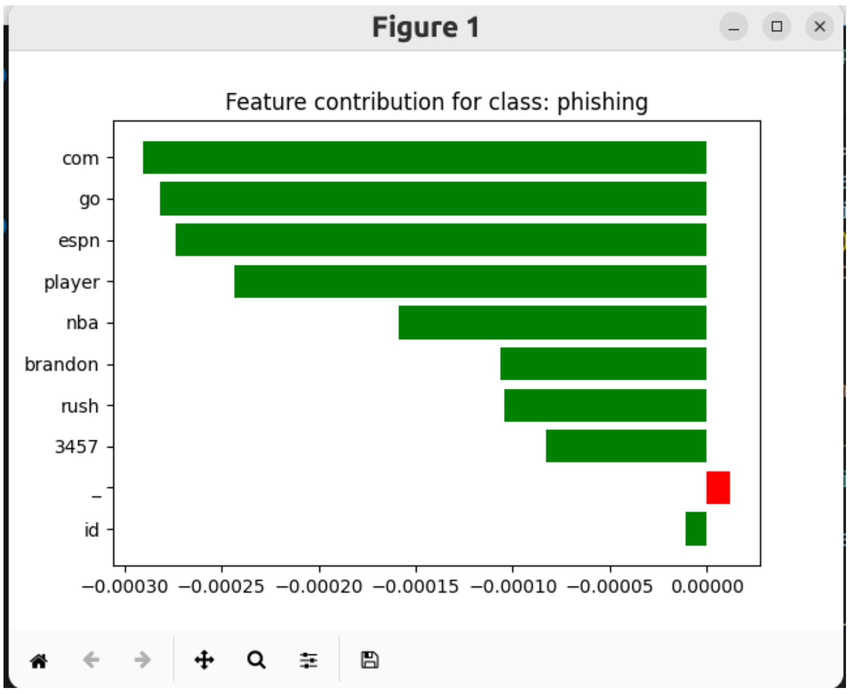


Figure 10: Feature Contribution To Phishing Class

Despite these challenges, Experiment 3 showcased the potential of utilizing an intermediate layer to extract features for XAI explanations. This step forward provided valuable insights into the model's decision-making process, opening avenues for further advancements in explainable AI for URL detection.

6.4 Experiment 4

In Experiment 4, the challenge of model bias was addressed by analyzing the dataset's composition and its impact on model training and interpretability. The initial dataset's class distribution was quite imbalanced: Benign: 428,103, Defacement: 96,457, Phishing: 94,111, Malware: 32,520

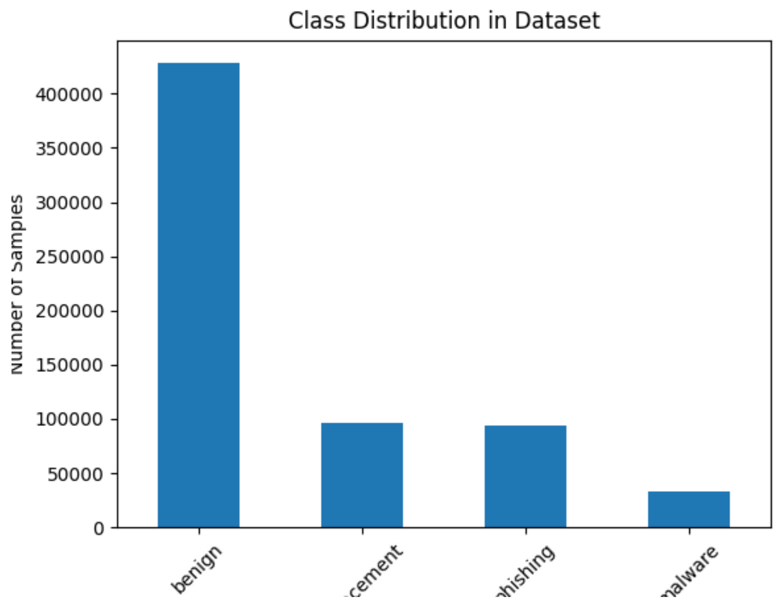


Figure 11: Total Dataset Distribution

Such an imbalance led to biases in the model towards the majority class, in this instance, 'benign,' potentially leading to the inadequate classification of minority class samples and a skewed perspective on feature importance. To correct this, I implemented a strategy to trim the dataset, downsampling the 'benign' class, resulting in a balanced distribution: Benign: 30,000, Malware: 30,000, Defacement: 30,000, Phishing: 30,000

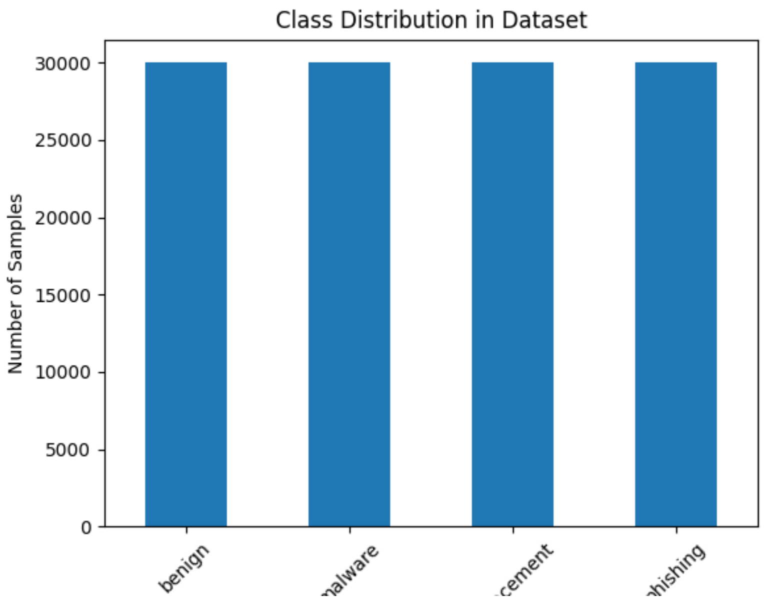


Figure 12: Balanced Dataset Distribution

This rebalanced dataset was then used to retrain the model over five epochs with a batch size of 64, ensuring that the model quickly adapted to the new data distribution without overfitting. Retraining a balanced dataset significantly reduced the model's bias. The explanations generated post-retraining showed a significant change in the XAI explanations provided compared to the previous ones.

In contrast with the experimentation from Experiment 3, the model trained with the balanced dataset showed a drastically different classification for the same URL. The 'phishing' class dominated here, with significant contributions from relevant features. The most notable thing was that out of 10 tests conducted using the same URLs in the previous experiment, only five were phishing, two were benign, and three were malware. This shift in feature importance highlights the model's improved ability to recognize and interpret different classes equitably.

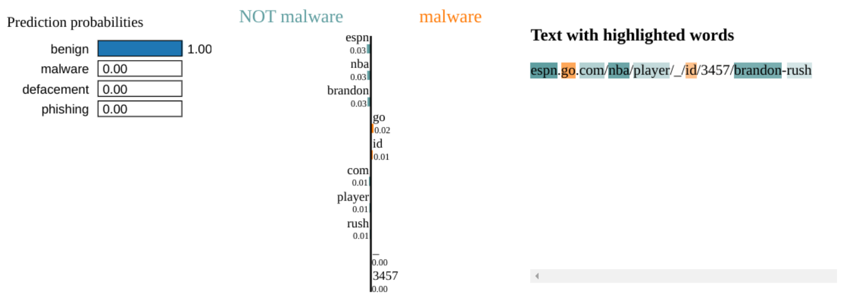


Figure 13: Lime Explanation Of URL From The Balanced Dataset Model

This is a more detailed representation of the malware class

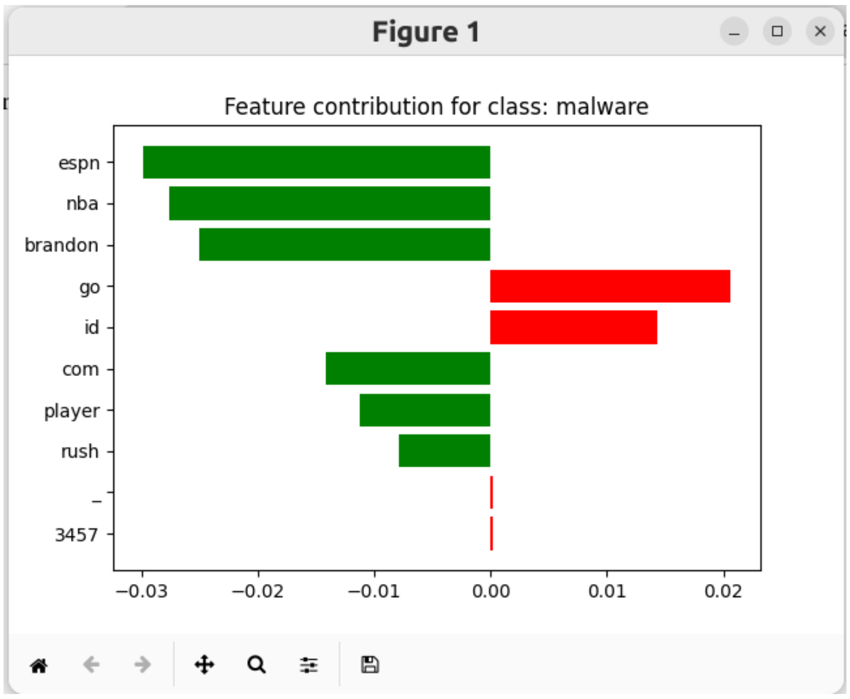


Figure 14: Feature Contribution To The Malware Class

7 Discussion

This chapter presents and analyzes the results obtained from the training of the deep convolutional neural network (DCNN) training and explainable AI (XAI) tool application. It delves into the interpretation of results, and the comparison of the model's performance before and after XAI tool integration. Finally, it discusses the implications of the findings for malicious URL detection in the cybersecurity landscape.

7.1 Interpretation of Results

The research presented in this study has yielded valuable insights into the effectiveness of XAI tools in explaining machine learning models, particularly dynamic convolutional neural networks (DCNN) for the purpose of URL classification. The findings demonstrate that XAI tools, particularly LIME, can provide helpful information about the features contributing to model predictions. However, the results also highlight the potential limitations of XAI tools, mainly when dealing with class imbalances in data. The research and the results deduced from XAI tools showed that using DCNNs in malicious URL detection can be very rewarding due to their accuracy in classification. Still, it also raises a challenge when addressing model biases during model training. In the experiments conducted above, it is evident that during model training, regardless of the group classification number of URLs provided, the model was still found to be biased. Moreover, it can be established that the main shortfalls with XAI tools were computational intensity and data biases, calling for more research in efficient processing and understanding of model bias in the training data.

7.2 Comparison of Model Performance Before and After XAI Tool Integration

Integrating XAI tools did not significantly impact the DCNN's classification accuracy. However, it is essential to note that XAI tools are computationally intensive, making them slower when integrated into dynamic convolutional neural networks. Regardless, the explanations provided by these tools provide valuable insights without compromising the model's performance.

7.3 The role of XAI tools in the cyber security community

The findings of this study hold significant implications for malicious URL detection in the domain of cybersecurity:

1. **Enhanced Understanding of Model Behavior:** XAI tools provide a deeper Improved Rule Refinement: Identifying key features through XAI can guide the refinement of threat detection rules, focusing on the most critical indicators of compromise and reducing false positives.

2. **Enhanced Incident Response:** XAI explanations can accelerate incident response by providing security analysts with a rapid understanding of the nature of an attack and the factors that triggered the model's alert.

3. **Building Trust and Transparency:** XAI fosters trust and transparency among security professionals, stakeholders, and the general public by explaining model decisions. This transparency is crucial for building confidence in the reliability of machine learning-driven cybersecurity systems.

7.4 Challenges and Limitations in Practical Implementation

Despite its transformative potential, XAI faces several challenges in its practical implementation within the cybersecurity domain:

1. **Computational Complexity:** XAI techniques can be computationally demanding, especially for large-scale machine learning models. This computational burden can limit their real-time applicability and require careful resource allocation.

2. **Data Availability:** The effectiveness of XAI hinges on the quality and availability of data. Insufficient or biased data can lead to misleading or inaccurate explanations, hindering the reliability of XAI insights.

3. **Human Interpretation:** Interpreting XAI explanations often requires machine learning and cybersecurity expertise. Security professionals may need more training to utilize XAI tools and derive meaningful insights effectively.

4. **Model Complexity:** As machine learning models become increasingly sophisticated, explaining their decision-making processes becomes more challenging. Developing new XAI techniques to keep pace with model advancements is essential for maintaining their effectiveness.

8 Conclusion and Future Work

In conclusion, this research has demonstrated the effectiveness of DCNNs in detecting malicious URLs, achieving an average prediction time of 0.3-0.5 seconds and a best-performing model accuracy of 96.7 percent. Applying XAI tools, particularly LIME, provided valuable insights into the features that contributed most significantly to the models' predictions. However, challenges arise when dealing with large and complex datasets as the computational demands of XAI tools increase, leading to slower explanation times and diminished classification performance.

To address these challenges, future research should focus on developing more efficient and scalable techniques for explaining deep convolutional neural networks (DCNNs). Promising avenues include exploring approximation techniques to reduce the computational overhead of explanations and investigating methods for parallelizing XAI computations. By enhancing the efficiency of XAI tools, researchers can empower researchers to understand and refine complex models, leading to improved malicious URL detection and enhanced cybersecurity.

References

Abad, S., Gholamy, H. and Aslani, M. (2023). Classification of Malicious URLs Using Machine Learning, *Sensors* **23**(18): 7760.
URL: *https://www.mdpi.com/1424-8220/23/18/7760*

Ali, S., Abuhmed, T., El-Sappagh, S., Muhammad, K., Alonso-Moral, J. M., Confalonieri, R., Guidotti, R., Del Ser, J., Díaz-Rodríguez, N. and Herrera, F. (2023). Explainable Artificial Intelligence (XAI): What we know and what is left to attain Trustworthy Artificial Intelligence, *Information Fusion* **99**: 101805.
URL: *https://www.sciencedirect.com/science/article/pii/S1566253523001148*

Alsaedi, M., Ghaleb, F. A., Saeed, F., Ahmad, J. and Alasli, M. (2022). Cyber Threat Intelligence-Based Malicious URL Detection Model Using Ensemble Learning, *Sensors* **22**(9): 3373. Number: 9 Publisher: Multidisciplinary Digital Publishing Institute.
URL: *https://www.mdpi.com/1424-8220/22/9/3373*

Chaudhuri, K. D. (2021). Intent Classification with Convolutional Neural Networks.
URL: *https://www.analyticsvidhya.com/blog/2021/12/intent-classification-with-convolutional-neural-networks/*

Chuang, Y.-N., Wang, G., Yang, F., Liu, Z., Cai, X., Du, M. and Hu, X. (2023). Efficient XAI Techniques: A Taxonomic Survey. arXiv:2302.03225 [cs].
URL: *http://arxiv.org/abs/2302.03225*

Ibrahim, R. and Shafiq, M. O. (2023). Explainable Convolutional Neural Networks: A Taxonomy, Review, and Future Directions, *ACM Computing Surveys* **55**(10): 206:1–206:37.
URL: *https://dl.acm.org/doi/10.1145/3563691*

Jang-Jaccard, J. and Nepal, S. (2014). A survey of emerging threats in cybersecurity, *Journal of Computer and System Sciences* **80**(5): 973–993.
URL: *https://www.sciencedirect.com/science/article/pii/S0022000014000178*

Jones, C. (2021). 50 Web Security Stats You Should Know In 2023.
URL: *https://expertinsights.com/insights/50-web-security-stats-you-should-know/*

Keita, Z. (2023). Explainable AI, LIME & SHAP for Model Interpretability | Unlocking AI's Decision-Making.
URL: *https://www.datacamp.com/tutorial/explainable-ai-understanding-and-trusting-machine-learning-models*

Lemay, D. J., Basnet, R. B. and Doleck, T. (2020). Examining the Relationship between Threat and Coping Appraisal in Phishing Detection among College Students, *Journal of Internet Services and Information Security* **10**(1): 38–49.
URL: *https://doi.org/10.22667/JISIS.2020.02.29.038*

Li, Y., Yang, Z., Chen, X., Yuan, H. and Liu, W. (2019). A stacking model using URL and HTML features for phishing webpage detection, *Future Generation Computer Systems* **94**: 27–39.
URL: *https://www.sciencedirect.com/science/article/pii/S0167739X1830503X*

Meske, C., Bunde, E., Schneider, J. and Gersch, M. (2022). Explainable Artificial Intelligence: Objectives, Stakeholders, and Future Research Opportunities, *Information Systems Management* **39**(1): 53–63. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/10580530.2020.1849465.
URL: *https://doi.org/10.1080/10580530.2020.1849465*

Mondal, D. K., Singh, B. C., Hu, H., Biswas, S., Alom, Z. and Azim, M. A. (2021). SeizeMaliciousURL: A novel learning approach to detect malicious URLs, *Journal of Information Security and Applications* **62**: 102967.
URL: *https://www.sciencedirect.com/science/article/pii/S2214212621001794*

Naresh, R., Gupta, A. and Giri, S. (2020). Malicious URL Detection System Using Combined SYM and Logistic Regression Model.
URL: *https://papers.ssrn.com/abstract=3598024*

Palmer (2019). Cybersecurity: 99% of email attacks rely on victims clicking links.
URL: *https://www.zdnet.com/article/cybersecurity-99-of-email-attacks-rely-on-victims-clicking-links/*

Patgiri, R., Biswas, A. and Nayak, S. (2023). deepBF: Malicious URL detection using learned Bloom Filter and evolutionary deep learning, *Computer Communications* **200**: 30–41.
URL: *https://www.sciencedirect.com/science/article/pii/S0140366422004832*

Rameem Zahra, S., Ahsan Chishti, M., Iqbal Baba, A. and Wu, F. (2022). Detecting Covid-19 chaos driven phishing/malicious URL attacks by a fuzzy logic and data mining based intelligence system, *Egyptian Informatics Journal* **23**(2): 197–214.
URL: *https://www.sciencedirect.com/science/article/pii/S1110866521000815*

Saleem Raja, A., Vinodini, R. and Kavitha, A. (2021). Lexical features based malicious URL detection using machine learning techniques, *Materials Today: Proceedings* **47**: 163–166.
URL: *https://www.sciencedirect.com/science/article/pii/S2214785321028947*

Sengupta, N. (2021). Natural Language Processing Using CNNs for Sentence Classification.
URL: *https://www.analyticsvidhya.com/blog/2021/09/natural-language-processing-using-cnns-for-sentence-classification/*

Sun, X. and Liu, Z. (2023). Domain generation algorithms detection with feature extraction and Domain Center construction, *PLoS One* **18**(1): e0279866.
URL: *https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9882890/*

Wang, Z., Ren, X., Li, S., Wang, B., Zhang, J. and Yang, T. (2021). A Malicious URL Detection Model Based on Convolutional Neural Network, *Security and Communication Networks* **2021**: e5518528. Publisher: Hindawi.
URL: *https://www.hindawi.com/journals/scn/2021/5518528/*

Wejinya, G. and Bhatia, S. (2021). Machine Learning for Malicious URL Detection, pp. 463–472.