DETECTION OF PHISHING WEBSITES USING DEEP LEARNING

A PROJECT REPORT

Submitted by

GAURI PRADEEP (TKM23MCA-2028)

to

TKM College of Engineering

Affiliated to

The APJ Abdul Kalam Technological University

In partial fulfilment of the requirements for the award of the degree of

MASTER OF COMPUTER APPLICATION

Thangal Kunju Musaliar College of Engineering
Kerala

DEPARTMENT OF COMPUTER APPLICATIONS

NOVEMBER 2024

DECLARATION

I undersigned hereby to declare that the project report on **DETECTION OF PHISHING WEBSITES USING DEEP LEARNING**, submitted for partial fulfilment of the requirements for the award of degree of Master of Computer Application of the APJ Abdul Kalam Technological University, Kerala is a Bonafide work doneby me under supervision of **Dr. Fousia M Shamsudeen.** This submission represents my ideas in my own words and where ideas or words of others have been included, we have adequately and accurately cited and referenced the original sources. I also declare that we have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data oridea or fact or source in our submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for theaward of any degree, diploma or similar title of any other University.

KOLLAM GAURI PRADEEP

11/11/24

DEPARTMENT OF COMPUTER APPLICATIONS

TKM COLLEGE OF ENGINEERING

(Government Aided and Autonomous)

KOLLAM - 5

CERTIFICATE

This is to certify that, the report entitled **Detection of Phishing websites using Deep learning** submitted by **Gauri Pradeep (TKM23MCA-2028)** to the **APJ Abdul Kalam Technological University** in partial fulfilment of the requirements for the award of the Degree of **Master of Computer Application** is a Bonafide record of the project work carried out by him/her under my/our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

----------------------------------   ----------------------------------

Internal Supervisor(s)   Mini Project Co-ordinator

ACKNOWLEDGEMENT

First and foremost, I thank GOD almighty and our parents for the success of this project. I owesincere gratitude and heart full thanks to everyone who shared their precious time and knowledge for the successful completion of my project.

I am extremely grateful to Prof. Natheera Beevi M, Head of the Department, Department of Computer Applications, for providing us with best facilities.

I would like to thank my project guide Dr. Fousia M Shamsudeen.

I am greatly obliged to my project co-ordinator, Prof. Sheera Shamsu

I profusely thank all other faculty members in the department and all other members of TKM College of Engineering, for their guidance and inspirations throughout my course of study.

I owe thanks to my friends and all others who have directly or indirectly helped me in the successful completion of this project.

GAURI PRADEEP

ABSTRACT

Phishing website detection is essential for enhancing cybersecurity by protecting users from fraudulent sites that seek to steal sensitive information. This project aims on building an effective phishing detection system using deep learning with meta-learning approach, which equip the model with the adaptability to recognize evolving phishing patterns. The CNN-BiLSTM model serves as the core architecture, chosen for its strengths in both feature extraction and sequence analysis, making it well-suited to understand the unique structures found in phishing URLs. Additionally, the Reptile meta-learning algorithm is incorporated to improve the model's ability to generalize across diverse types of phishing URLs.

This project introduces a meta-learning-based CNN-BiLSTM model specifically designed for phishing website detection. By using the Reptile meta-learning algorithm, our approach aims to enhance the model's ability to adapt to a variety of phishing patterns, which is essential given the diversity and continual evolution of phishing tactics. The CNN layers in the model capture character-level features within URLs, while BiLSTM layers process the sequential patterns that often signal phishing attempts. During training, rigorous optimization techniques are employed, such as dropout and batch normalization, to prevent overfitting and enhance model robustness, allowing for stable performance even with unseen data. To make this solution accessible and user-friendly, a Flask web application, Phishing Shield, has been developed, allowing users to test URLs and check their legitimacy in real-time through a straightforward interface. This project ultimately aims to contribute to cybersecurity by offering a scalable and efficient tool capable of adapting to new phishing threats as they emerge, providing an extra layer of defence against increasingly sophisticated phishing attacks.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Phishing attacks have grown more complex and deceptive in recent years, with cybercriminals constantly refining their tactics to trick individuals into disclosing sensitive personal information, such as usernames, passwords, and financial data. These attacks often come in the form of fraudulent emails, websites, or messages that appear legitimate, manipulating users into making harmful decisions without realizing the threat. As digital technologies continue to evolve, so too do the strategies employed by these malicious actors, making it increasingly difficult for traditional security measures to keep pace. Conventional security systems, which rely on signature-based detection or predefined patterns, are often unable to recognize new or sophisticated phishing techniques, leading to vulnerabilities that can be exploited by attackers. The scale and variety of phishing attacks further exacerbate the problem. Cybercriminals are no longer limited to simple email scams but have expanded their reach across multiple channels, including social media, SMS (smishing), and even voice-based phishing (vishing). Additionally, phishing campaigns have become more targeted, with attackers using social engineering tactics to craft messages tailored to specific individuals or organizations. These highly personalized approaches make it even harder for users to identify phishing attempts, as they appear more credible and relevant. Furthermore, phishing attacks are not only a personal threat but also a significant concern for organizations and businesses. They can lead to substantial financial losses, data breaches, and reputational damage. Phishing can also serve as a gateway for more severe forms of cybercrime, such as ransomware attacks, where malicious actors encrypt company data and demand payment for its release. As these attacks become more sophisticated and widespread, the risk to both individuals and businesses intensifies, making the need for effective detection and prevention methods more urgent than ever. Recently, machine learning and deep learning models have emerged as powerful tools in phishing detection, offering automated, scalable solutions that can analyze large datasets of URLs or email content to identify patterns associated with phishing activities. Existing systems for phishing detection are primarily deployed in cybersecurity tools to protect users from malicious websites that attempt to steal sensitive information. These systems often rely on traditional machine learning methods, where URLs are classified based on predefined patterns and feature engineering techniques. Typically, these frameworks involve intensive preprocessing and feature extraction, requiring a substantial amount of manual intervention to achieve reasonable accuracy. This results in significant time and resource demands for both data preparation and model training, making these systems challenging to scale and adapt quickly to new phishing tactics. In conventional approaches, extensive datasets and manually crafted features are required to train models, which then attempt to detect phishing sites based on previously observed patterns. However, these systems often struggle to keep up with evolving phishing strategies and variations in URL structures. Moreover, they may face limitations in accuracy, which reduces their reliability. While traditional phishing detection methods can deliver acceptable results under controlled conditions, they are generally time-consuming to develop and maintain. The reliance on intensive feature engineering and manual updates to adapt to new threats can be a barrier to widespread deployment. Additionally, these systems may lack the flexibility to generalize across diverse phishing types, limiting their effectiveness in real-world environments where phishing tactics are constantly evolving.The Phishing Website Detection System offers a robust solution for real-time identification and classification of phishing websites, essential for safeguarding users from malicious sites. Traditional phishing detection systems often require extensive manual feature engineering and complex pipelines to achieve accurate classification, making them time-consuming and labor-intensive to develop and maintain. These systems may struggle to keep up with constantly evolving phishing tactics, necessitating frequent adjustments to stay effective in recognizing new threats. Unlike traditional detection methods, the proposed Phishing Detection System uses deep learning and meta-learning techniques, specifically a CNN-BiLSTM model combined with the Reptile algorithm, to streamline and automate the detection process. By incorporating advanced neural network architectures, the system minimizes the need for manual feature engineering and adapts quickly to diverse phishing patterns. This adaptability allows the system to deliver immediate and reliable results, which are crucial for real-time phishing prevention. This innovative approach has the potential to make phishing detection more efficient, accessible, and effective, providing a modern solution to combat the growing threats in cybersecurity.

This project focuses on developing a cutting-edge phishing website detection system by integrating a hybrid deep learning model that combines Convolutional Neural Networks (CNN), Bidirectional Long Short-Term Memory (BiLSTM) networks, and meta-learning techniques. The CNN is designed to identify patterns and anomalies in the temporal structure of URLs, capturing crucial features that distinguish phishing sites from legitimate ones. Meanwhile, the BiLSTM network enhances detection accuracy by understanding the spatial dependencies within the URL, analyzing it from both left-to-right and right-to-left directions to capture contextual relevance. By incorporating meta-learning through the Reptile algorithm, the system adapts quickly to new phishing tactics, enabling it to generalize across various datasets with minimal retraining. This combination of advanced neural network architectures and meta-learning provides a scalable, efficient, and robust solution that improves phishing detection performance while minimizing manual intervention and ensuring adaptability to constantly evolving phishing techniques.

CHAPTER 2

LITERATURE REVIEW

Phishing website detection has been an area of intense research, with various methodologies proposed to improve the identification of fraudulent sites. Siva et al. [1] explored a machine learning-based approach using URL features, primarily leveraging Decision Trees (DT) to classify phishing websites. This approach showed promise with high accuracy through metrics such as precision, recall, and specificity. However, the study acknowledged the challenges in detecting sophisticated phishing techniques that mimic legitimate URLs, highlighting the need for further research in feature engineering to improve accuracy. Moving beyond URL feature-based detection, Opara et al. [2] introduced the HTMLPhish model, which applies Convolutional Neural Networks (CNNs) directly to HTML content for phishing detection. While CNNs are effective in extracting temporal features, they lack the ability to combine these features across layers, thus limiting the model's capacity to capture more complex, interdependent patterns. The study suggested that periodic retraining would be necessary to adapt to evolving phishing tactics, underlining the importance of adaptive and integrated feature extraction methods. To tackle the integration of both spatial and sequential data, Zonyfar et al. [3] proposed a hybrid HCNN-LSTM model, combining CNNs and Long Short-Term Memory (LSTM) networks to predict legitimate websites by processing 87 extracted URL attributes. This model demonstrated a classification accuracy of 95.19%, emphasizing the benefit of integrating both spatial and sequential data. However, the study pointed out challenges in handling large datasets, advocating for further model optimization to enhance scalability and real-time performance. In a similar vein, but with a focus on real-time phishing detection, Tang and Mahmoud [4] developed a framework using an RNN-GRU model implemented as a browser extension. Their model achieved an impressive accuracy of 99.18%, surpassing traditional machine learning methods like SVM and Logistic Regression. It emphasized the need for quick adaptation to new phishing attacks. However, the study highlighted the computationally intensive nature of real-time processing, suggesting the need for efficient feature selection techniques to balance accuracy and response time. In a different approach, Banik et al. [5] proposed an ensemble learning method based on LSTM networks for phishing URL detection, achieving over 99.5% accuracy with a low false positive rate below 0.15%. While this approach outperformed several machine learning methods, the authors noted that LSTM networks primarily capture sequential patterns and focus on spatial features without combined feature extraction, which could limit the model's ability to detect more complex patterns. Similarly, Mambina et al. [6] introduced a hybrid CNN-LSTM-LSTM model for SMS spam detection in the Swahili language, emphasizing its versatility by utilizing CNN layers for feature extraction and LSTM layers for sequential pattern recognition. However, the study acknowledged limitations such as a small dataset size and lack of computational efficiency analysis, suggesting the need for further exploration of real-world applications and cross-domain adaptability. Collectively, these studies underscore the importance of combining deep learning techniques, task-specific feature extraction, and adaptive training methods to improve phishing detection accuracy while addressing challenges related to scalability, real-time performance, and dataset diversity. Together, these studies emphasize the critical role of integrating advanced deep learning techniques, domain-specific feature extraction, and adaptive training strategies to enhance the accuracy of phishing website detection. The findings highlight the need for models that not only excel in recognizing complex patterns but also adapt dynamically to evolving threats. This integration can significantly improve the model's robustness and generalization across diverse datasets, ensuring its effectiveness in various real-world scenarios. Additionally, these approaches must address key challenges such as scalability, which ensures models can handle large and diverse datasets efficiently, and real-time performance, which is crucial for deploying systems that can identify phishing attempts swiftly. Furthermore, the importance of dataset diversity is clear, as training on varied and representative data enhances the model's ability to recognize phishing sites across different types of attacks, domains, and languages. These collective insights suggest that a multi-faceted approach, combining powerful learning algorithms with smart feature engineering and adaptable training techniques, is essential for developing more accurate and resilient phishing detection systems.

CHAPTER 3

METHODOLOGY

3.1 Objective

To develop a phishing website detection system by combining a hybrid CNN-BiLSTM model for combined feature extraction, incorporating meta-learning to enhance adaptability across diverse phishing attacks, optimizing model performance through hyperparameter tuning and advanced techniques like dropout, regularization, and batch normalization, and enabling real-time detection of phishing websites by dynamically processing and classifying URLs with high accuracy and minimal delay.

3.2 Proposed Methodology

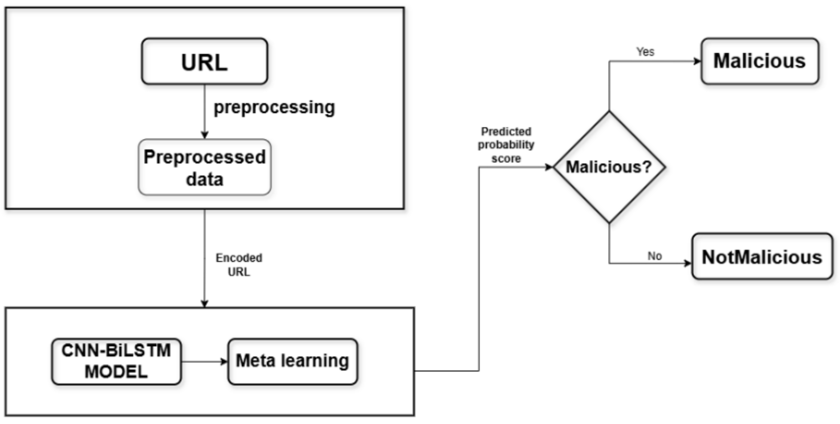


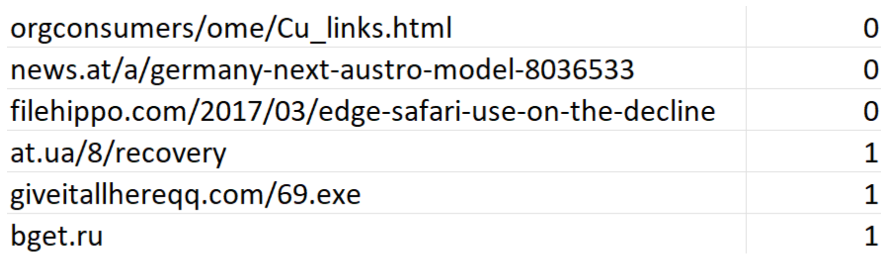
Fig 3.1: Block Diagram

The methodology of the work as shown in Fig.1 follows a structured approach to accurately classify URL as phishing or legitimate. First, data acquisition involves collecting a comprehensive dataset of approximately 140,000 URLs, representing both phishing and legitimate websites Following data collection, preprocessing is performed to clean and prepare the URLs, ensuring consistency and quality for effective feature extraction. This processed dataset serves as the foundation for training the model. A meta-learning-based hybrid model, combining Convolutional Neural Networks (CNN) for feature extraction and Bidirectional

Long Short-Term Memory (BiLSTM) networks for sequential data processing, is developed to enhance phishing detection. Finally, performance evaluation is conducted to assess the model's accuracy and robustness, providing insights into its effectiveness in identifying phishing URLs and its adaptability to emerging threats. The proposed **Phishing Website Detection System** processes and analyses URLs to accurately classify them as either phishing or legitimate using a hybrid deep learning model that combines a CNN-BiLSTM architecture with the Reptile meta-learning algorithm. The system captures and processes input data from real-world environments, analysing URLs to classify them as phishing or legitimate using a meta-learning-based CNN-BiLSTM model. The CNN component extracts temporal features from the URL structure, identifying patterns that differentiate phishing sites from legitimate ones. The BiLSTM layer enhances the model by extracting spatial features, capturing contextual relationships within the URL’s structure. By incorporating the Reptile meta-learning algorithm, the system quickly adapts to new phishing tactics, improving classification accuracy with minimal retraining. This approach ensures efficient, real-time phishing detection with high adaptability to evolving threats.This phishing detection system is built for easy integration as a web application using a Flask-based interface. Users can quickly submit URLs and receive instant feedback on whether they are legitimate or phishing. With the Reptile meta-learning algorithm, the system is adaptable to new phishing tactics, providing a scalable and efficient cybersecurity solution.

3.3 Datasets Used

The success of this phishing detection system relies on a well-curated dataset of URLs labelled as phishing or legitimate. The dataset used in this project is a publicly available and contains a diverse range of URLs to ensure the model can identify phishing sites across different techniques and tactics. The dataset for phishing website detection was downloaded from Kaggle, containing labelled URLs as malicious (phishing) or non-malicious (legitimate). The dataset includes two primary classes: malicious (phishing) and non-malicious (legitimate) websites. The dataset consists of over 140,000 URLs, with each URL labelled as either phishing (1) or legitimate (0). The URLs have been pre-processed for model input, including tokenization and padding for consistent input length.



orgconsumers/ome/Cu_links.html	0
news.at/a/germany-next-austro-model-8036533	0
filehippo.com/2017/03/edge-safari-use-on-the-decline	0
at.ua/8/recovery	1
giveitallhereqq.com/69.exe	1
bget.ru	1

Fig 3.2: Sample URLs from dataset

3.4 Techniques Used

3.4.1 Data Preprocessing

Data preprocessing was essential to prepare the URLs for optimal model performance. The following steps were taken:

- Tokenization: Each URL was converted into a list of integer tokens based on characters contained in the "printable" list.
- Padding/Truncation: URLs were truncated to a maximum length of 75 characters or padded with zeros if shorter to ensure uniform input length.
- Label Extraction: Labels (isMalicious) are extracted into a numpy array for classification.

3.4.2 CNN-BiLSTM Hybrid Model

This project utilizes a hybrid deep learning architecture that combines a one-dimensional Convolutional Neural Network (1D CNN) with Bidirectional Long Short-Term Memory (BiLSTM) layers to effectively detect and classify phishing URLs. To further enhance performance, the Reptile meta-learning algorithm is integrated, which strengthens the model's ability to generalize across diverse phishing attack patterns, including those it hasn't previously encountered. By merging the 1D CNN and BiLSTM architectures, the model leverages the advantages of both approaches, resulting in a more resilient and adaptable system for phishing URL detection.

The architecture consists of:

- Convolutional Layers: Capture local patterns in URL characters, such as sequences and structures.

- Pooling Layers: Reduce the dimensionality of the extracted features while preserving important information.
- BiLSTM Layers: Allows the model to learn contextual information from both past and future inputs, improving its ability to understand the entire sequence of data, which is crucial for tasks like URL classification where the order of characters or tokens plays an important role in identifying malicious patterns.

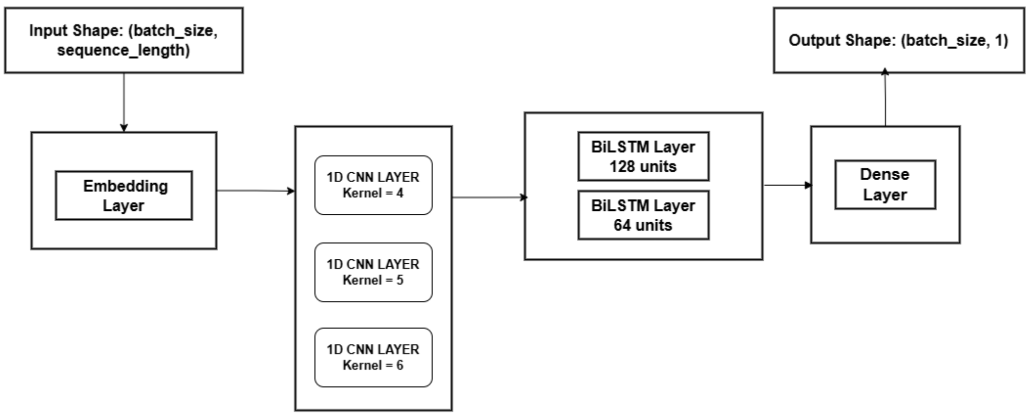


Fig 3.3: CNN-BiLSTM Architecture

The CNN component of the model is designed to capture local patterns and structures within the URL characters, such as recurring sequences and specific character groupings that may signify a phishing attempt. These convolutional layers help the model identify key features in the URL that are indicative of malicious intent, even if these features appear at various locations within the string. The pooling layers that follow the convolutional layers serve to reduce the dimensionality of the extracted features. By doing so, they simplify the data while still retaining the most important information, which is crucial for preventing the loss of essential features during the learning process.

On the other hand, the BiLSTM layers of the model are responsible for learning contextual dependencies across the entire sequence of URL characters. Unlike traditional LSTMs, which process data in a unidirectional manner (either forward or backward), BiLSTMs analyze data in both directions. This bidirectional approach allows the model to capture important contextual information from both the preceding and succeeding characters, significantly improving its ability to understand the meaning of the URL as a whole. This is particularly valuable in URL classification tasks, where the order and relationship between characters can make a significant difference in identifying whether a URL is legitimate or phishing. By combining the spatial feature extraction power of CNNs with the sequential understanding of BiLSTMs, this hybrid model provides a powerful tool for detecting malicious URLs in a highly efficient and accurate manner.

3.4.3 Reptile Algorithm

The Reptile algorithm is a meta-learning algorithm that enhances the adaptability and generalization of the CNN-BiLSTM model to new, previously unseen phishing and legitimate URLs. Meta-learning, often referred to as "learning to learn," focuses on training models that can efficiently adapt to new tasks with limited data. Reptile, in particular, improves the model's ability to generalize across various tasks, which is essential in phishing detection, where new types of phishing URLs constantly emerge. Below is a more detailed explanation of the key steps involved in the Reptile algorithm for meta-learning:

Task Sampling

At each iteration of the Reptile algorithm, tasks are randomly selected from the dataset. These tasks represent subsets of the data and expose the model to a diverse set of URL patterns and characteristics. By randomly sampling tasks, the model learns to recognize a broad range of phishing and legitimate URLs, simulating various environments or scenarios in which phishing attempts could occur. This helps ensure that the model doesn't become overly specialized to one particular set of URLs but instead gains the flexibility to identify different types of phishing attempts.

Inner Loop

After the random tasks are chosen, the model enters the inner loop, where it is trained on each selected task for several steps. During this phase, the model fine-tunes its parameters to adapt to the specific characteristics of each task's data. The inner loop is key to enabling the model to adjust and learn from the distinct features of different phishing or legitimate URLs. This process allows the model to become more flexible and responsive to variations in the data, which is particularly important given the constant evolution of phishing techniques.

Weight Averaging

After the model has trained on all the tasks, the learned weights from each task are averaged. This averaging process creates a set of generalized model parameters that reflect the knowledge gained from all the tasks. Weight averaging is a critical step, as it ensures that the model avoids overfitting to any one specific task and instead learns more generalizable representations. These averaged weights help the model apply its knowledge across a broader range of phishing and legitimate URL patterns, making it more capable of handling new, unseen tasks with high accuracy.

Outer Loop

Once the weights have been averaged, the updated model is used to adjust the original CNN-BiLSTM model. The outer loop updates the model with the averaged weights, allowing it to generalize more effectively across multiple tasks. This ensures that the improvements made during the inner loop, which focused on specific tasks, are incorporated into the model's broader capacity to adapt. The outer loop is responsible for fine-tuning the model so that it can handle a wide variety of phishing attempts without becoming too specialized.

Iterative Updates

This process is repeated iteratively over several cycles. Each iteration improves the model's ability to generalize by building on the insights gained from the previous iteration. Through continuous task sampling, model adaptation in the inner loop, weight averaging, and outer-loop updates, the model gradually becomes more effective at recognizing new phishing patterns. These iterative updates are essential for maintaining and enhancing the model's performance, allowing it to adapt to emerging phishing techniques and remain effective in an ever-changing online landscape.

In summary, the Reptile algorithm's process of task sampling, inner-loop adaptation, weight averaging, and iterative updates significantly boosts the CNN-BiLSTM model's ability to generalize across various phishing patterns. This meta-learning approach enables the phishing detection system to be more adaptable, efficient, and capable of recognizing previously unseen phishing attacks, making it a powerful tool in combating online threats.

3.4.4 Hyperparameter Tuning with KerasTuner

Before finalizing the CNN-BiLSTM model, hyperparameter tuning was conducted to identify the optimal configuration that would maximize model performance. The goal of this process was to select the best hyperparameters that enhance the model's generalization ability, ensuring robust performance on unseen data during testing. KerasTuner was employed to automate the search for the best hyperparameters, focusing on key parameters such as the learning rate and batch size. The learning rate, which controls how quickly the model updates its weights, was tuned using a logarithmic scale between 1×10^{-2} and 1×10^{-3} while the batch size was adjusted for optimal stability and convergence. The tuning process involved task creation, where various subsets of the data were simulated to represent different types of phishing and legitimate URLs, improving the model's ability to generalize across diverse patterns. The RandomSearch method from KerasTuner was then used to explore different hyperparameter combinations over multiple trials to determine the best-performing configuration. Additionally, the model was trained using the Reptile meta-learning algorithm, which enhances generalization by performing task-specific training and weight averaging across tasks. A custom callback, HistorySaver, was implemented to save the model's training history, including accuracy, loss, validation accuracy, and validation loss for each trial, which was stored in CSV files for easy analysis. After selecting the optimal hyperparameters based on validation accuracy, the final model was trained for a fixed number of epochs (20) and saved for future deployment and use.

Hyperparameter tuning plays a crucial role in optimizing the CNN-BiLSTM model by fine-tuning key parameters such as learning rate and batch size to maximize performance on validation data, ensuring better generalization to new, unseen data. The integration of the Reptile meta-learning algorithm further enhances the model's adaptability, enabling it to perform effectively across various URL patterns. A custom HistorySaver callback is employed to track and save all relevant metrics during training, allowing for easy analysis and selection of the best-performing configurations. Finally, the model is saved for future use in real-time URL phishing detection applications, ensuring that it is well-tuned and capable of delivering high performance across diverse URL patterns.

3.4 Performance metrices

In the domain of cybersecurity, accurately classifying URLs as either “phishing” or “legitimate” is crucial to protecting users from online threats. This project primarily evaluates the performance of the CNN-BiLSTM model using accuracy as the main metric, alongside precision, recall, and F1-score for a comprehensive understanding of model performance.

Accuracy is the primary metric used to assess the model, representing the proportion of correct predictions—both phishing and legitimate—that the model makes. In the context of phishing detection, high accuracy indicates that the model is effective at distinguishing between phishing and legitimate URLs, ensuring reliable classification across the dataset.

Precision measures the proportion of URLs predicted as phishing that were actually phishing, focusing on minimizing false positives, which represent legitimate URLs mistakenly flagged as phishing. High precision ensures the model is precise in its phishing alerts, reducing the likelihood of false alarms.

Recall quantifies the model's ability to identify actual phishing URLs, minimizing false negatives, which are undetected phishing URLs. In phishing detection, recall is crucial as missing even a few phishing URLs can pose significant security risks.

The F1-score, which combines precision and recall, offers a balanced assessment of the model’s performance. Although accuracy is prioritized as the primary metric, the F1-score provides additional insight into the model’s ability to handle both false positives and false negatives, particularly valuable in cases where phishing URLs are less frequent in the dataset.

Metric Formulas

The key metrics used in this project are as follows:

$$Accuracy = \frac{TP+FN}{TP+FN+TN+FP} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(3.4.1)$$

$$Precision = \frac{TP}{TP+FP} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(3.4.2)$$

$$Recall = \frac{TP}{TP+FN} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(3.4.3)$$

$$F1score = 2 * \frac{Precision*Recall}{Precision+Recall} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(3.4.4)$$

In summary, accuracy serves as the primary evaluation metric for assessing the CNN-BiLSTM model’s performance in phishing URL classification, supported by precision, recall, and F1-score for a more detailed analysis.

CHAPTER 4

EXPERIMENTAL RESULTS, DISCUSSION AND ANALYSIS

4.1 Environmental setup

This phishing website detection project integrates a wide range of tools and libraries, each playing a vital role in the journey from initial model development to the final deployment of a web application. At the heart of the project's development environment is Google Colab, a cloud-based platform that offers free access to GPUs and TPUs, allowing for significantly accelerated computations, which are crucial for training deep learning models like the CNN-BiLSTM architecture. Colab's cloud infrastructure provides a responsive setup for experimenting with code, adjusting model parameters, and performing various optimizations. Integrated with Google Drive, it enables seamless file saving and retrieval, making collaboration, model versioning, and data management efficient and straightforward. These advantages make Colab an excellent platform for testing different architectures and refining models, which is essential in machine learning tasks that demand high processing power.

In addition to Colab, VS Code (Visual Studio Code) is a core tool for coding, debugging, and managing different project files. As a powerful code editor, VS Code supports multiple languages and includes features like an integrated terminal, Git support, and plugins for Python and Jupyter notebooks. This versatility makes it ideal for a project that combines various elements such as model training scripts, Flask application code, and data preprocessing scripts. VS Code's rich debugging capabilities allow quick identification and resolution of issues within the code, contributing to a smoother development cycle. Additionally, its extensive plugin support allows the integration of various tools that are helpful for machine learning projects, such as the Python extension, which supports code linting, auto-completions, and syntax highlighting, improving code readability and reducing errors.

The project is primarily written in Python, a language highly regarded in the machine learning community for its simplicity, readability, and the breadth of libraries available. Python's ecosystem includes a vast array of tools and libraries specifically designed for tasks such as data processing, machine learning, and deployment, making it the perfect choice for a project of this scale. Python's integration capabilities with TensorFlow, Keras, and Flask facilitate a cohesive flow from data preprocessing to model training and finally to deployment in a web application, ensuring that the model's lifecycle is entirely streamlined within a single language.

To ensure the web application functions correctly and provides users with a smooth experience, Google Chrome is used to test the Flask app interface. Chrome's Developer Tools offer a robust suite for inspecting elements, diagnosing layout issues, and testing the functionality of JavaScript code within the app. By testing the application in Chrome, developers can ensure that URLs submitted for phishing detection are processed correctly, and predictions are displayed accurately, with any bugs in the front end swiftly identified and resolved.

Several libraries empower the machine learning and meta-learning components of this project. TensorFlow serves as the foundational deep learning framework, handling the backend computation for the CNN-BiLSTM model. TensorFlow's flexibility supports CPU and GPU computation, which is essential for efficiently managing and training deep learning models. Paired with Keras, TensorFlow allows for the simplified configuration of neural network layers, loss functions, and optimizers, providing a more accessible API that supports rapid experimentation. This streamlined development process helps developers fine-tune and adjust the model, contributing to improved accuracy and faster convergence.

NumPy and Pandas handle data processing and transformation tasks. NumPy enables the efficient handling of large datasets through its support for multi-dimensional arrays and complex mathematical operations, which are essential for working with the high-dimensional data generated during model training. Pandas, on the other hand, excels in data manipulation and management. It simplifies data preparation through its DataFrame structure, which is highly efficient for data loading, cleaning, and transformation—key steps in getting the data ready for model input. Together, NumPy and Pandas form a robust toolkit for managing and transforming data, ensuring it's in the optimal format for training.

For implementing a meta-learning approach, the learn2learn library is employed, introducing adaptability to the model by enabling it to generalize to new tasks rapidly. Meta-learning, specifically through algorithms like Reptile, is leveraged to make the CNN-BiLSTM model more robust in detecting phishing patterns, enhancing its ability to generalize across different types of phishing data. This adaptability provides the model with an added layer of learning, allowing it to detect subtle variations in phishing sites that may not have been seen in the training set, thus improving its real-world applicability.

KerasTuner is used for hyperparameter tuning, a critical component in optimizing model performance. KerasTuner automates the search for the most effective configuration of parameters, such as the learning rate, batch size, and the number of layers. Through this library, the project benefits from an efficient exploration of different hyperparameter spaces, which allows for fine-tuning that leads to better model performance without extensive manual experimentation. This automated process contributes to a more reliable and accurate model, enhancing its performance in phishing detection.

Finally, Flask is the lightweight web framework used to deploy the model as an interactive web service. With Flask, users can easily input URLs to determine whether they are legitimate or phishing sites, receiving predictions in real time. Flask's ability to support RESTful APIs allows for seamless integration between the backend model and the user interface, creating a smooth user experience. Lightweight yet versatile, Flask makes it easy to develop a user-friendly interface, ensuring that the model's phishing detection capabilities are accessible to users who may not have a technical background.

Through this combination of tools, libraries, and frameworks, the project achieves a cohesive and efficient workflow, from data preprocessing and model training to real-time deployment in a user-friendly web application. This robust setup ensures high performance and adaptability, providing an effective solution for detecting phishing websites.

4.2 Results

MODEL	ACCURACY	PRECISION	RECALL	F1SCORE
Meta-learning Based CNN-BiLSTM	**97%**	**97%**	**97%**	**97%**

Table 4.1: Performance Scores of the CNN-BiLSTM Model in Phishing Website Dataset

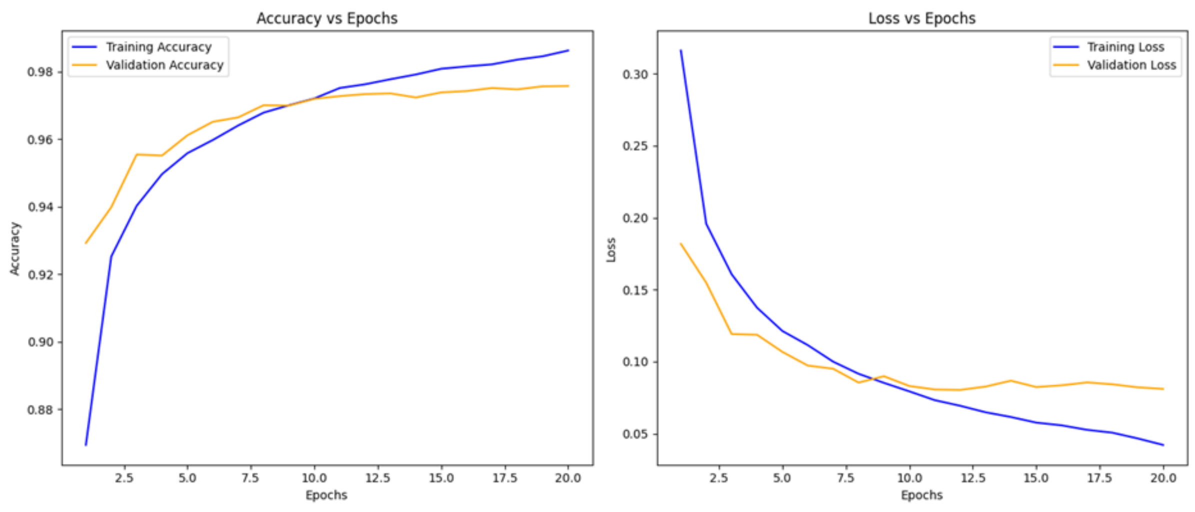


Fig 4.1: Accuracy and Loss Curve

In this study, the **meta-learning-based CNN-BiLSTM model** was evaluated for phishing website detection, specifically for its ability to distinguish between phishing and legitimate URLs. The model achieved an impressive **validation accuracy of 97.57%** and a **test accuracy of 97.47%**, indicating its strong performance in correctly classifying phishing websites. These high accuracy rates reflect the model's ability to generalize well from the training data, suggesting that it effectively learned the underlying patterns that differentiate phishing URLs from legitimate ones.The model's performance can be further visualized through the **accuracy graphs** presented in **Figure 4.1**. These graphs illustrate the steady increase in both validation and test accuracy over the epochs, with only minor fluctuations, indicating that the model consistently improved its classification ability as it was trained. The gradual rise in accuracy reflects the model's capacity to not only memorize the training examples but also to generalize its learning to new, unseen data. This is crucial for a real-world application like phishing website detection, where the model must accurately predict future phishing attempts.The **validation loss** and **test loss** graphs in **Figure 4.1** provide additional insights into the model's training dynamics. Both graphs show a consistent decrease in loss over the training epochs, which indicates that the model was progressively minimizing its error during training. The steady reduction in loss suggests that the model is learning effectively and avoiding overfitting, a common challenge in deep learning models. As the model continues to minimize the loss, its performance improves, as evidenced by the rising accuracy.

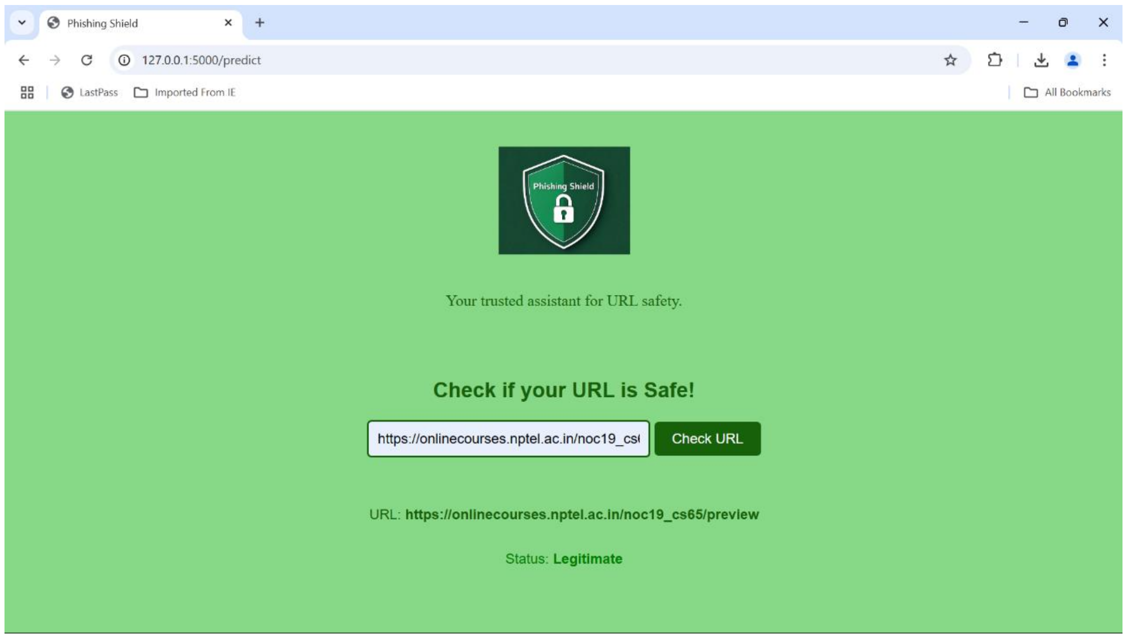
To evaluate the model comprehensively, we also calculated **precision** and **recall** metrics. The **meta-learning-based CNN-BiLSTM model** achieved a **precision of 97%** and a **recall of 97%**. These results indicate that the model is highly effective at both identifying phishing websites (high recall) and ensuring that legitimate websites are not mistakenly flagged as phishing (high precision). Precision measures the proportion of true positive predictions among all positive predictions, while recall measures the model's ability to identify all the actual phishing websites. In the context of phishing detection, both high precision and recall are critical, as they ensure that phishing sites are detected with minimal false positives, and legitimate sites are not misclassified as threats.The impressive performance of this model demonstrates the effectiveness of integrating meta-learning with a CNN-BiLSTM architecture for phishing website detection. Meta-learning enhances the model's ability to learn better representations of the data, which contributes to the high precision and recall rates observed. By using meta-learning, the model can adapt to the complexities of the phishing detection task, leading to a more robust and reliable solution.

In summary, the **meta-learning-based CNN-BiLSTM model** stands out as a highly efficient and effective tool for phishing website detection. With its high accuracy, precision, and recall, the model is well-suited for real-world applications where accurate and timely detection of phishing websites is crucial. The results underscore the potential of meta-learning in improving deep learning models for complex tasks such as phishing detection, highlighting its importance in the development of advanced security systems.

4.3 Prediction

In the field of cybersecurity, the model developed in this project—a meta-learning-based CNN-BiLSTM—plays a crucial role in accurately classifying URLs as either phishing or legitimate. The model is trained to detect phishing websites based on patterns and characteristics found within the URLs. After thorough training and fine-tuning, the model is tested to predict the legitimacy of a URL and classify it accordingly. The predictions are stored in a CSV file, which serves as a comprehensive record for further analysis.The CSV file contains detailed information on whether a given URL has been classified as "phishing" or "legitimate." This structured output facilitates further analysis and decision-making, enabling continuous improvement of the model's performance. Security professionals can use these predictions to develop strategies for phishing website detection, enhancing both automated detection and manual intervention processes. This project demonstrates the practical application of deep learning and meta-learning models in cybersecurity, specifically for phishing website detection. The use of advanced techniques for URL classification provides a sophisticated solution to the growing problem of phishing attacks. The CSV file also serves as a useful tool for tracking model performance, allowing informed decisions and optimization based on real-world data.

To enhance accessibility, the meta-learning-based CNN-BiLSTM model is integrated into a Flask web application. The user-friendly interface allows users to simply paste a URL and receive a prediction on whether the URL is phishing or legitimate. This web application bridges the gap between advanced model predictions and user needs, making phishing detection more accessible. The Flask web app is hosted and can be easily accessed via any web browser. Users can input a URL, and the app communicates with the trained model to return a prediction. This real-time prediction capability makes the app a valuable tool for individuals who may come across suspicious URLs while browsing or engaging in online activities. When a URL is submitted, the Flask app preprocesses it in the same way as during the model’s training, including tokenization and padding. The processed URL is then passed into the CNN-BiLSTM model, which classifies it as either "phishing" or "legitimate." The result is displayed immediately on the web interface, providing users with actionable insights for protecting against phishing attacks. The system benefits from the power of the meta-learning-based model, ensuring accurate and reliable predictions. As the model continues to improve with further training, its predictions will become even more effective in real-world scenarios. The Flask web app also enables the tool to be easily adopted by security-conscious individuals or organizations. In addition to real-time predictions, the web app can be further enhanced with features like logging prediction data for analysis and providing users with detailed reports on phishing trends. These future enhancements will support the evolution of the project, allowing improvements in user experience and system performance. The integration of the meta-learning-based CNN-BiLSTM model into the Flask web application provides a powerful, user-friendly tool for detecting phishing websites. By analyzing and classifying URLs, the system offers real-time predictions that can help users avoid phishing attacks and enhance online security practices.



Your trusted assistant for URL safety.

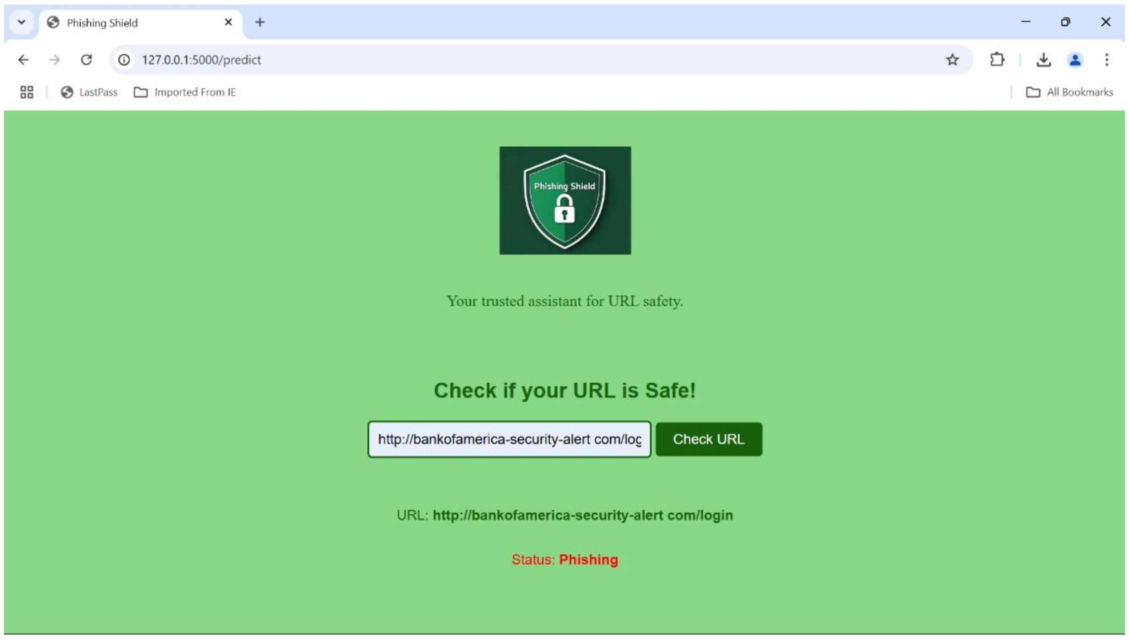
Check if your URL is Safe!

https://onlinecourses.nptel.ac.in/noc19_cs(

URL: **https://onlinecourses.nptel.ac.in/noc19_cs65/preview**

Status: **Legitimate**

Fig 4.2: Screenshot of prediction (Legitimate)



Your trusted assistant for URL safety.

Check if your URL is Safe!

http://bankofamerica-security-alert com/log

URL: **http://bankofamerica-security-alert com/login**

Status: **Phishing**

Fig 4.3: Screenshot of prediction (Phishing)

In conclusion, the meta-learning-based BiLSTM-CNN model for phishing website detection has proven to be an effective tool in identifying phishing URLs with high accuracy. While the model performs well in its current state, ongoing research and optimizations could further improve its speed and adaptability for broader, real-time deployment. The system holds significant potential for enhancing online security and can be integrated into various applications, providing a robust defence against phishing attacks.

CHAPTER 5

CONCLUSION AND FUTURE SCOPE

The Phishing Website Detection System effectively met its goal of distinguishing between phishing and legitimate websites using a meta-learning-based BiLSTM-CNN model. This approach showed strong performance in accurately identifying malicious websites, making it a valuable tool in combating the increasing threat of phishing. By offering an efficient and reliable method for detecting phishing sites, the system significantly contributes to online security by helping users avoid fraud and security breaches. The growing prevalence of phishing attacks makes this technology a crucial asset in the cybersecurity field, providing an essential layer of protection for users navigating the digital landscape.

Looking ahead, there are several opportunities for improvement and further development of the system. One potential enhancement is the integration of the detection system into browser extensions, which would allow real-time phishing detection directly within the user's browsing experience. By flagging suspicious URLs as users navigate websites, this integration would enhance the system's accessibility and convenience. Additionally, exploring more sophisticated meta-learning techniques, such as MAML or ProtoNet, could improve the model's ability to generalize across a broader range of phishing tactics, enhancing its robustness and performance in diverse environments. These improvements would help the system adapt more quickly to emerging attack vectors, ensuring that it remains a powerful tool for safeguarding users' online security. Further enhancements to the system could focus on adapting the model to account for region-specific phishing tactics and language variations in phishing URLs. Phishing techniques can vary widely across regions and cultures, with attacks targeting local businesses or government websites that are more prominent in certain areas. Additionally, phishing websites may use local languages or cultural references to deceive users. By training the model to recognize region-specific patterns and language variations, the system could more effectively identify phishing attempts tailored to a particular audience. This would be especially useful in multilingual settings, improving the model’s global applicability and enhancing its detection capabilities. Another avenue for improvement is extending the model’s functionality to classify the type of phishing attack. Currently, the model focuses on distinguishing between phishing and legitimate websites, but phishing attacks can vary widely in their objectives. For example, credential phishing attempts to steal login information, financial scams try to deceive users into transferring money, and malware delivery phishing tricks users into downloading harmful software. By classifying the type of phishing attack, the model could provide users with more granular insights into the threats they are facing, allowing for more targeted responses. This classification could be especially valuable for security professionals and organizations, who could prioritize certain types of threats and tailor their security measures accordingly. Optimizing the model's performance for low-resource environments is another key avenue for improvement. Many users rely on mobile devices or low-powered embedded systems that may not have the computational resources to run resource-intensive models effectively. Optimizing the model to perform efficiently on such devices would expand its reach and make phishing detection accessible to a much broader audience. This could involve reducing the model's memory and processing requirements through techniques like model pruning, quantization, or knowledge distillation. By making the system accessible on devices with limited computing power, it could protect a wider range of individuals and organizations, particularly in regions with limited access to high-performance hardware.

These enhancements, combined with ongoing advancements in meta-learning, will further strengthen the phishing detection system's capabilities, ensuring its continued relevance in the fight against the ever-evolving landscape of online threats.

CHAPTER 6

REFERENCES

[1] **V Samba Siva, K Meghana Reddy, G Likith Sai, P Mahesh, V Madhuri** (2024). Phishing Website Detection Based on URL Using Machine Learning , International Journal of Innovative Research in Science, Engineering and Technology. Volume 13, Issue 3.

[2] **Chidimma Opara, Bo Wei, Yingke Chen** (2020). HTMLPhish: Enabling Phishing Web Page Detection by Applying Deep Learning Techniques on HTML Analysis, 2020 International Joint Conference on Neural Networks *(IJCNN)*. IEEE

[3**] Candra Zonyfar, Jung-Been Lee, Jeong-Dong Kim** (2023). HCNN-LSTM: Hybrid Convolutional Neural Network with Long Short-Term Memory Integrated for Legitimate Web Prediction, Journal of Web Engineering, Vol. 22_5, 757–782

[4] **Tang, L., & Mahmoud, Q. H.** (2021). A deep learning-based framework for phishing website detection. *IEEE Access*, 9, 107110–107123

[5] **Bireswar Banik and Abhijit Sarma**. (2023). Phishing Url Detection Using Lstm Based Ensemble Learning Approaches, International Journal of Computer Networks & Communications (IJCNC) Vol.15, No.1

[6] **Mambina, I. S., Michael, K. F., Ndibwile, J. M., & Mpuhwe, D**. (2021). Uncovering SMS spam in Swahili text using deep learning approaches. *IEEE Access*, 9, 12345-12358.