

AUTOMATED DETECTION OF MALICIOUS URLS USING DEEP LEARNING

¹SREE LAXMI KARUPATI, ²DR.P.JOEL JOSEPHSON

¹Student, Department of Electronics and Communication Engineering, Malla Reddy Engineering College(Autonomous).

Srilaxmikarupati09@gmail.com

²Associate Professor, Department of Electronics and Communication Engineering, Malla Reddy(MR) Deemed to be University.

ABSTRACT:

Phishing attacks have become a serious cybersecurity challenge as attackers continuously create deceptive URLs designed to imitate legitimate websites and obtain sensitive information such as usernames, passwords, financial credentials, and personal data. Conventional phishing detection techniques, including blacklists and manually defined rules, may have difficulty identifying newly generated and previously unseen malicious URLs. To address this challenge, this project presents PhishDetect, a hybrid deep learning approach for intelligent phishing URL classification. The proposed system analyzes the sequential characteristics of URLs and automatically learns patterns associated with phishing and legitimate web addresses. The methodology includes dataset uploading, data cleaning, URL tokenization, numerical encoding, dataset splitting, model training, performance evaluation, and URL prediction. Two bidirectional recurrent neural network architectures, Bidirectional Long Short-Term Memory (Bi-LSTM) and Bidirectional Gated Recurrent Unit (Bi-GRU), are incorporated to capture contextual information from URL sequences in both forward and backward directions. Bi-LSTM is utilized to learn long-term dependencies, while Bi-GRU provides efficient sequential representation with comparatively lower computational complexity. Their complementary learning capabilities are combined to develop a robust hybrid detection framework. The system evaluates classification performance using accuracy, precision, recall, and F1-score and provides a user-friendly web interface for practical testing and prediction. The proposed approach is designed to improve the identification of suspicious URLs while reducing dependence on manually engineered features. Overall, PhishDetect provides an automated and adaptable framework for phishing URL detection that can support safer online interactions and strengthen cybersecurity protection against evolving phishing threats.

Keywords: Phishing Detection, Phishing URLs, Deep Learning, Bi-LSTM, Bi-GRU, Hybrid Model, URL Classification, Cybersecurity, Sequence Learning, Malicious URL Detection.

Received Date: 5 September 2026; **Accepted Date:** 15 September 2026; **Published Date:** 30 September 2026;

This is an open-access article distributed under the terms of the Creative Commons Attribution 4.0 International License (CC BY 4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author(s) and the source are properly cited.

I. INTRODUCTION

The rapid expansion of Internet-based services has transformed the way individuals and organizations communicate, conduct financial transactions, access information, and perform business activities. However, this increased dependence on online platforms has also created opportunities for cybercriminals to exploit users through phishing attacks. Phishing is a form of cyberattack in which malicious actors create deceptive websites, messages, or URLs that imitate legitimate services to

persuade users to disclose sensitive information such as usernames, passwords, banking credentials, and personal details. URL-based phishing detection has therefore become an important cybersecurity problem. Early detection systems commonly relied on blacklists, where URLs were compared against previously identified malicious addresses. Although blacklist-based approaches can be simple and fast, they are inherently limited because newly generated or previously unseen phishing URLs may not yet exist in the

database. Machine-learning approaches subsequently introduced the use of URL characteristics and manually engineered features to improve classification. Studies have demonstrated that URL structure, lexical characteristics, domain information, and suspicious patterns can provide useful indicators for identifying malicious websites [1], [4]. However, traditional feature-based approaches may require substantial feature engineering and may not fully capture the sequential relationships present within URL strings. Sahingoz *et al.* demonstrated the effectiveness of machine-learning techniques for URL-based phishing detection, showing that automated classification can achieve strong performance while supporting detection of new websites [12]. These developments highlight the need for intelligent detection methods capable of learning complex URL representations directly from data rather than depending exclusively on predefined rules or manually selected features.

Deep learning has emerged as an effective direction for malicious URL and phishing detection because neural networks can learn complex representations from large volumes of sequential and textual data. Conventional machine-learning techniques often depend on handcrafted lexical or structural characteristics, whereas deep learning can learn useful representations directly from URL strings. URLNet, for example, introduced an end-to-end deep-learning framework for learning URL representations from character- and word-level information, addressing limitations associated with manual feature engineering and unseen URL characteristics [9]. Recurrent neural networks are particularly suitable for this problem because URLs can be represented as ordered sequences of characters or tokens. Long Short-Term Memory (LSTM) networks were developed to address difficulties associated with learning long-term dependencies in sequential data [22]. Bidirectional LSTM extends this concept by processing sequences in both forward and backward directions, enabling the network to obtain contextual information from both sides of an input sequence [23]. Research on malicious URL classification has demonstrated the effectiveness of LSTM and Bi-LSTM architectures for identifying phishing-related patterns. In particular, comparative

experiments involving LSTM, Bi-LSTM, and GRU reported strong performance for recurrent architectures, with Bi-LSTM achieving particularly high classification performance [24]. GRU provides another recurrent architecture with a comparatively simpler gating structure and can learn sequential dependencies efficiently [24]. These findings indicate that recurrent deep-learning models are well suited to phishing URL analysis because suspicious characteristics may appear at different positions throughout a URL.

Based on these developments, the proposed PhishDetect: A Hybrid Deep Learning Approach focuses on combining the complementary capabilities of Bidirectional Long Short-Term Memory (Bi-LSTM) and Bidirectional Gated Recurrent Unit (Bi-GRU) networks for phishing URL classification. The motivation for using a hybrid architecture is that different recurrent models can capture different aspects of sequential URL information. Bi-LSTM is capable of learning long-term contextual dependencies, while Bi-GRU provides efficient sequential processing with a comparatively simpler recurrent structure. Previous research has shown that deep recurrent models can effectively identify malicious URL patterns, while hybrid deep-learning architectures can combine different representation-learning capabilities to improve phishing detection [8], [9]. Research specifically investigating Bi-GRU has also demonstrated its usefulness for malicious URL classification, particularly when contextual dependencies within URL sequences are important [6]. The proposed system therefore processes URL data through preprocessing, tokenization, numerical encoding, and dataset partitioning before applying the deep-learning architecture. The learned representations are used to classify URLs into phishing and legitimate categories. Model performance is evaluated using accuracy, precision, recall, and F1-score to provide a balanced assessment of classification effectiveness. The overall objective is to develop an automated phishing detection framework that reduces dependence on manual feature engineering and provides a practical foundation for identifying suspicious URLs. Such a system can potentially support cybersecurity applications such as browser protection, email security, online transaction monitoring, and threat-intelligence systems. The proposed approach builds upon established

URL-based machine learning and deep-learning research while exploring the combined use of Bi-LSTM and Bi-GRU for robust sequential URL analysis.

II SURVEY OF RESEARCH

1. Traditional and Blacklist-Based Phishing Detection

Early phishing detection systems primarily depended on blacklists and predefined rules to identify malicious websites. In these approaches, a URL is compared against a database containing previously reported or confirmed phishing addresses. Such techniques are relatively simple and can provide fast decisions when a malicious URL is already known. However, their major limitation is their dependence on previously collected information. Newly generated phishing URLs, modified domains, and previously unseen attacks may not be present in the blacklist, allowing them to evade detection. Research on malicious URL detection has therefore emphasized the need to move beyond static blacklists toward data-driven classification methods [1]. Subsequent studies explored URL characteristics and machine-learning techniques to identify suspicious patterns directly from URLs [4], [12]. Systematic reviews have confirmed that blacklist, heuristic, machine-learning, and deep-learning approaches represent different stages in the development of phishing detection systems [3]. These limitations provide the motivation for intelligent approaches that can learn patterns from URL data rather than relying only on previously identified threats. The proposed PhishDetect system addresses this limitation by using deep learning to learn sequential URL representations automatically.

2. Machine Learning-Based Phishing URL Detection

Machine-learning techniques have significantly improved phishing detection by allowing systems to classify URLs according to their characteristics. Researchers have investigated features such as URL length, special characters, domain structure, suspicious keywords, and other lexical properties to distinguish malicious URLs from legitimate ones. Sahingoz *et al.* demonstrated the usefulness of machine-learning algorithms for URL-based phishing

detection, showing that automated classification can effectively identify phishing websites [12]. Feature-rich systems such as CANTINA+ also demonstrated how multiple URL and webpage characteristics can be incorporated into machine-learning-based phishing detection [4]. However, traditional machine-learning approaches generally depend on feature extraction and feature engineering, which can require domain expertise and may fail to capture complex relationships within URL strings. A systematic review of phishing detection techniques reported that machine learning has been extensively investigated, with Random Forest being among the commonly used classifiers [3]. Although these methods can achieve strong classification performance, the increasing complexity of phishing attacks creates a requirement for models capable of automatically learning representations from raw URL sequences. Therefore, the transition from manually engineered features toward representation-learning methods forms an important foundation for the development of PhishDetect.

3. Deep Learning for Phishing Detection

Deep learning has become an important research direction for phishing detection because neural networks can automatically learn complex patterns from large datasets. Unlike conventional machine-learning methods that often depend on manually selected features, deep-learning models can learn useful representations during training. A systematic literature review covering 43 research studies found that URL-related information was the most frequently used data source for deep-learning-based phishing detection. The review also identified Deep Neural Networks, Convolutional Neural Networks, and Recurrent Neural Networks/LSTM as widely used architectures [25]. Importantly, the study reported that hybrid deep-learning approaches demonstrated strong performance compared with several individual architectures [25]. Another comprehensive review of phishing URL detection research similarly identified CNN, RNN, LSTM, and hybrid architectures as major approaches for analyzing URL sequences [26]. URLNet further demonstrated the value of learning URL representations directly from character- and word-level information rather than relying entirely on handcrafted features [9]. These findings establish that deep learning

is suitable for the proposed system because phishing URLs contain structural and sequential patterns that can be learned automatically. PhishDetect builds upon this research direction by applying recurrent deep-learning architectures to learn contextual information from URL sequences.

4. LSTM and Bi-LSTM for Sequential URL Analysis

Long Short-Term Memory networks are particularly relevant to phishing URL detection because URLs can be represented as ordered sequences of characters or tokens. LSTM was introduced to address the difficulty of learning long-term dependencies in conventional recurrent neural networks [22]. Its memory and gating mechanisms allow relevant information to be retained across longer sequences, making it useful for sequential classification tasks. Bidirectional LSTM extends this capability by processing an input sequence in both forward and backward directions. This enables the model to consider contextual information from different portions of the URL simultaneously. Research on phishing URL classification has demonstrated the usefulness of recurrent architectures for learning URL patterns [8]. More recent research has also investigated Bi-LSTM directly for phishing URL classification, demonstrating its ability to model sequential URL information [27]. Reviews of deep-learning-based phishing detection consistently identify LSTM and recurrent neural networks as important approaches for URL analysis [25]. The use of Bi-LSTM in PhishDetect is therefore motivated by its ability to capture contextual dependencies that may occur at different positions within a URL. Instead of depending entirely on manually defined URL characteristics, the model can learn sequential representations during training and use them for phishing-versus-legitimate classification.

5. GRU and Bi-GRU-Based Sequential Learning

Gated Recurrent Units provide another recurrent architecture for processing sequential information. GRU was introduced as a gated recurrent architecture designed to learn sequence dependencies while maintaining a comparatively simpler structure than LSTM [24]. Its gating mechanism controls the information that should be retained or discarded

during sequence processing. This makes GRU attractive for applications where sequential patterns must be learned efficiently. In phishing detection, URL strings contain ordered character and token patterns that can provide useful evidence about whether an address is malicious. Research on recurrent neural networks has demonstrated the suitability of gated architectures for sequence modeling [24], while broader phishing-detection reviews identify recurrent models as an important category of deep-learning techniques [25]. Bi-GRU extends the standard GRU by processing the sequence in two directions, allowing contextual information from both sides of the URL to contribute to classification. The use of Bi-GRU in PhishDetect is intended to complement Bi-LSTM by providing an additional representation of the URL sequence. Combining the two bidirectional recurrent architectures can potentially capture different aspects of URL structure while maintaining an end-to-end learning process. This approach is consistent with the broader research trend toward hybrid deep-learning architectures for improving phishing detection performance.

6. Hybrid Deep Learning Models and Research Gap

Recent research increasingly investigates hybrid deep-learning models because combining architectures can provide complementary representation-learning capabilities. A systematic review of deep-learning approaches for phishing detection found that hybrid models were among the approaches providing strong performance, although deep-learning systems also face challenges related to computational requirements, overfitting, interpretability, and real-time detection [25]. Research has explored combinations such as CNN-LSTM and Bi-LSTM-CNN for learning different aspects of URL information [25]. More recent studies have also investigated sophisticated architectures combining Bi-LSTM with attention and convolutional components to improve phishing URL classification [28]. Despite these advances, the literature indicates continuing challenges involving zero-day attacks, adversarial manipulation, model generalization, explainability, and real-time deployment [26]. These gaps motivate the development of a comparatively focused hybrid architecture using Bi-LSTM and Bi-GRU. The

proposed PhishDetect system combines the two bidirectional recurrent networks to learn complementary sequential representations of URLs and classify them as phishing or legitimate. The model is evaluated using accuracy, precision, recall, and F1-score, providing multiple perspectives on classification quality. Thus, the proposed work extends existing sequential deep-learning research by investigating a hybrid Bi-LSTM and Bi-GRU framework specifically for automated phishing URL detection.

III. PROPOSED SYSTEM

The proposed PhishDetect system is designed as an automated phishing URL detection framework that uses a hybrid deep-learning architecture based on Bidirectional Long Short-Term Memory (Bi-LSTM) and Bidirectional Gated Recurrent Unit (Bi-GRU) networks. The main purpose of the system is to distinguish between phishing and legitimate URLs by learning meaningful patterns directly from URL sequences. Unlike conventional approaches that depend heavily on blacklists or manually selected URL characteristics, the proposed system performs automated representation learning from the input data. The complete workflow begins with uploading a labelled URL dataset containing phishing and legitimate samples. The uploaded data is then examined and prepared for subsequent processing. The system performs preprocessing to remove unsuitable or inconsistent records and converts URL strings into numerical representations that can be supplied to the neural networks. This provides a standardized input format for the learning process. The proposed framework is intended to address the limitations of static phishing detection approaches, particularly their difficulty in recognizing previously unseen malicious URLs. The overall design follows a modular structure consisting of dataset upload, preprocessing, dataset splitting, model training, evaluation, and prediction components.

After preprocessing, the dataset is divided into training and testing subsets. The training portion is used to develop the deep-learning models, while the unseen testing portion is reserved for evaluating their classification capability. The proposed architecture uses Bi-LSTM and Bi-GRU as its principal learning components. Bi-LSTM processes URL sequences in both forward and backward directions, allowing the model to capture relationships between different portions of the URL. Bi-GRU follows a similar bidirectional strategy while providing a comparatively lightweight recurrent structure. The two models are intended to learn complementary characteristics from the same URL representation. Their learned information is then combined to create the proposed hybrid classification framework. The hybrid approach is expected to provide a richer representation of URL patterns than relying on a single recurrent architecture. The system ultimately produces a binary classification indicating whether the supplied URL belongs to the phishing or legitimate category. The project document specifically identifies Bi-LSTM as useful for learning long-term sequential dependencies and Bi-GRU as an efficient sequential-learning component.

The proposed system also incorporates a performance-evaluation component to assess the effectiveness of the trained model. The testing data is used to calculate accuracy, precision, recall, and F1-score, enabling the system to evaluate classification performance from multiple perspectives rather than relying on a single metric. After training, the prediction module allows a user to provide a new URL and obtain a classification from the trained hybrid model. The complete framework can be implemented as a web-based application using Python and Django, with HTML and CSS providing the user interface. This design makes the system accessible through a structured application workflow rather than limiting the model to an isolated experimental environment. The proposed system therefore provides an integrated pipeline covering data preparation, deep-learning model development, performance assessment, and practical URL prediction. Its primary contribution is the application of a hybrid Bi-LSTM and Bi-GRU architecture for learning sequential URL patterns and supporting automated phishing detection.

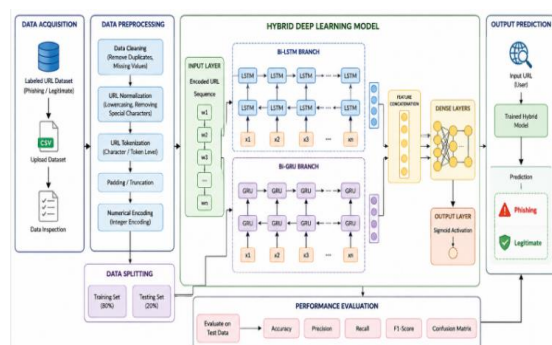


Fig1 : Proposed Architecture

IV. WORKING METHODOLOGY

The working methodology of PhishDetect: A Hybrid Deep Learning Approach is designed as a sequential pipeline that converts raw URL data into meaningful numerical representations and uses a hybrid Bi-LSTM and Bi-GRU architecture to classify URLs as phishing or legitimate. The methodology consists of dataset acquisition, preprocessing, sequence representation, dataset splitting, hybrid model development, training, evaluation, and final prediction. The proposed workflow is intended to reduce dependence on manually engineered URL features and enable the model to learn contextual and sequential characteristics directly from URL data. The system also provides a structured application workflow through which datasets can be uploaded, processed, models can be trained, and new URLs can be classified. The major stages of the methodology are described below.

1. Dataset Collection and Data Preparation

The first stage involves collecting a labelled dataset containing URLs and their corresponding classes. Each record represents a URL and its associated category, such as phishing or legitimate. The dataset is uploaded into the system through the dataset-upload module. Before model development, the data is inspected to identify missing values, duplicate records, inconsistent entries, and invalid URL samples. Removing unsuitable records helps improve the reliability of the subsequent learning process. The project methodology uses labelled URL data because supervised learning requires known target classes during model training. The prepared dataset therefore provides the foundation for both model development and performance evaluation. The system is designed to accept a dataset in a

suitable tabular format and validate its structure before proceeding to subsequent stages. The uploaded data is then passed to the preprocessing module for transformation into a representation that can be consumed by the deep-learning models. This dataset-upload and preparation process corresponds to the project's Dataset Upload Module.

2. URL Preprocessing and Numerical Representation

Raw URLs cannot be directly supplied to Bi-LSTM and Bi-GRU networks. Therefore, the URL strings are first cleaned and transformed into numerical sequences. The preprocessing stage removes unnecessary or inconsistent information and prepares URLs for tokenization. Each URL is divided into smaller units, such as individual characters or tokens, and each unit is assigned a numerical representation. The resulting sequence allows the neural network to process the URL as ordered sequential information. Because different URLs may have different lengths, padding or truncation can be applied to create a consistent input size. The general transformation can be represented as:

$$URL \rightarrow Tokens \rightarrow Numerical Sequence$$

For a URL represented as $(U=(u_1, u_2, \dots, u_n))$, the tokenization process produces a corresponding numerical sequence:

$$X = (x_1, x_2, \dots, x_n)$$

where (x_i) represents the numerical encoding of the corresponding URL character or token. This representation enables the proposed recurrent networks to identify relationships between different portions of a URL. The project document specifically identifies tokenization and encoding as important preprocessing operations for converting URLs into numerical sequences suitable for deep learning.

3. Dataset Splitting

After preprocessing, the complete dataset is divided into training and testing subsets. The training subset is used for learning the parameters of the proposed models, while the testing subset remains unseen during training and is used to estimate the model's classification performance. The methodology

described in the project uses an 80:20 training-testing division. Thus, approximately 80% of the prepared records are allocated to model development and 20% are retained for final evaluation.

The splitting process can be expressed as:

$$D = D_{train} \cup D_{test}$$

with

$$D_{train} = 80\%, \quad D_{test} = 20\%$$

Keeping the testing data separate is important because the final evaluation should measure how effectively the trained model classifies data that was not directly used during learning.

4. Hybrid Bi-LSTM and Bi-GRU Model Development

The central component of PhishDetect is the hybrid deep-learning model. The encoded URL sequence is supplied to two bidirectional recurrent branches: Bi-LSTM and Bi-GRU. Bi-LSTM processes the sequence from both directions, allowing the network to capture contextual relationships throughout the URL. It is particularly useful for learning dependencies that may occur between URL elements separated by several positions. Bi-GRU also processes the sequence bidirectionally but uses the comparatively simpler GRU architecture. The project identifies Bi-LSTM as a model for capturing long-term dependencies and Bi-GRU as an efficient sequential-learning component. The outputs generated by the two branches are combined to form a joint representation:

$$H_{hybrid} = H_{Bi-LSTM} \oplus H_{Bi-GRU}$$

where ($\oplus$) represents the combination or concatenation of the learned representations. The combined representation is then passed through fully connected layers before reaching the final classification layer. For binary classification, the output probability can be represented using the sigmoid function:

$$P(y = 1) = \frac{1}{1 + e^{-z}}$$

where (z) represents the output of the final neural-network layer. A probability above the selected classification threshold is assigned to the phishing class, while the alternative class is assigned as legitimate.

5. Model Training and Performance Evaluation

During training, the hybrid model learns from the labelled training dataset by adjusting its internal parameters to reduce classification error. The model repeatedly processes URL sequences and compares its predicted class with the known class label. After training, the testing dataset is supplied to the model to determine its generalization capability. The system evaluates performance using accuracy, precision, recall, and F1-score, which are specifically identified in the project methodology.

Accuracy is calculated as:

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

Precision is given by:

$$Precision = \frac{TP}{TP + FP}$$

Recall is calculated as:

$$Recall = \frac{TP}{TP + FN}$$

The F1-score combines precision and recall:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Here, (TP), (TN), (FP), and (FN) represent true positives, true negatives, false positives, and false negatives, respectively. These measures provide a broader assessment of phishing classification performance than accuracy alone.

6. URL Prediction and Application Workflow

After successful training and evaluation, the trained hybrid model is used for prediction. A user can enter a new URL through the application interface. The system applies the same preprocessing and numerical encoding procedure used during training to ensure that the new URL has a compatible representation. The processed URL is then passed to the trained Bi-LSTM and Bi-GRU branches, whose combined representation is evaluated by the classification layer. The final output identifies the URL as either phishing or legitimate. The project defines this as the Prediction Module, where new URLs can be tested using the trained hybrid model.

The complete working methodology can therefore be summarized as:

Dataset Upload → Data Cleaning → URL Tokenization → Numerical Encoding → 80:20 Data Split → Bi-LSTM + Bi-GRU Processing

→ Feature Combination → Classification → Performance Evaluation → New URL Prediction.

This integrated methodology provides PhishDetect with a complete pipeline from raw labelled URL data to practical phishing classification.

V. IMPLIMENTATION

The implementation of PhishDetect: A Hybrid Deep Learning Approach focuses on converting the proposed methodology into a functional phishing URL detection application. The system is developed using Python for data processing and deep-learning operations, while Django is used to provide the web-based application environment. HTML and CSS are used to design the user interface through which users can upload datasets and submit URLs for prediction. The implementation follows a sequential process beginning with dataset collection and validation, followed by data preprocessing, URL encoding, dataset splitting, hybrid model development, training, evaluation, and final prediction. The project specification identifies Python as the programming language, Django as the application framework, and HTML and CSS as the frontend technologies. The first stage of implementation is the development of the dataset upload facility. The system accepts a labelled dataset containing URLs and their corresponding classes. Each URL is associated with a category representing either a phishing or legitimate address. When the user uploads the dataset through the Django interface, the backend receives the file and verifies whether it contains the required information. The validation stage is important because the subsequent machine-learning operations depend on the availability of correctly formatted URL and label data. Records that are incomplete, inconsistent, or unsuitable for processing can be identified during this stage. The original project description specifies that the Dataset Upload Module is responsible for loading the phishing URL dataset and checking whether its format is compatible with the system.

After the dataset has been successfully uploaded, it is passed to the preprocessing component. Raw URLs cannot be directly supplied to the proposed recurrent neural

networks because the models require numerical input. Therefore, the URL information is cleaned and transformed into a machine-readable representation. Unnecessary or inconsistent records are removed, and the remaining URL strings are prepared for tokenization. Tokenization divides the URL into smaller units that can be represented numerically. Depending on the implementation, the URL can be represented at character level or token level. Each resulting unit is assigned a numerical value, allowing the complete URL to be represented as a sequence of numbers. The project document identifies cleaning, tokenization, and encoding as the main operations performed in the preprocessing module. Since URLs may contain different numbers of characters or tokens, the sequences are converted into a consistent format before they are supplied to the neural networks. Padding can be used for shorter sequences, while excessively long sequences can be restricted to the selected maximum sequence length. This produces uniform input data for the deep-learning architecture. The same preprocessing procedure must be applied to both the training data and new URLs submitted for prediction. Maintaining consistent preprocessing prevents differences between the data used during model development and the data received during actual prediction.

Once preprocessing is completed, the dataset is divided into training and testing portions. The project uses an 80:20 division, with approximately 80 percent of the processed records used for training and the remaining 20 percent reserved for testing. The training portion allows the hybrid model to learn the characteristics associated with phishing and legitimate URLs, whereas the testing portion is kept separate for evaluating the trained model. This separation is important because evaluating the model using the same records used for training would not provide a reliable indication of its ability to classify unseen URLs. The central implementation stage is the construction of the hybrid deep-learning model. The system uses two bidirectional recurrent architectures, namely Bi-LSTM and Bi-GRU. The Bi-LSTM branch receives the encoded URL sequence and processes it in both forward and backward directions. This enables the network to obtain information from different parts of the URL sequence and identify relationships that may not be apparent when the sequence is processed

in only one direction. The project describes Bi-LSTM as an architecture capable of capturing contextual information from both earlier and later portions of a URL.

The second branch uses Bi-GRU. GRU is a recurrent architecture designed to process sequential information while using a comparatively simpler gating mechanism than LSTM. The project explains that GRU uses fewer parameters and can provide efficient sequential learning. The bidirectional version processes the URL in both directions, allowing contextual information from the complete sequence to contribute to the learned representation. In the proposed implementation, the Bi-GRU branch works alongside the Bi-LSTM branch rather than replacing it. The outputs generated by the two branches are then combined to form a hybrid representation of the URL. The purpose of this combination is to utilize the complementary characteristics learned by the two recurrent architectures. Bi-LSTM is intended to capture long-term sequential dependencies, while Bi-GRU provides efficient learning of sequential patterns. The project describes this combination as the central feature of the proposed hybrid model and states that the combined architecture is intended to learn more complex phishing URL patterns. The combined representation is subsequently passed to the classification component, which determines the probability associated with the target classes. The resulting probability is compared with the selected classification threshold. Based on this decision, the URL is categorized as phishing or legitimate. The model therefore converts the learned sequential characteristics into a direct classification result.

After the architecture is constructed, the model is trained using the training dataset. During training, the encoded URL sequences are supplied to the Bi-LSTM and Bi-GRU branches together with their known class labels. The model compares its predicted output with the actual label and adjusts its internal parameters during successive training iterations. This process allows the network to gradually learn characteristics that distinguish phishing URLs from legitimate URLs. The project identifies model training as a separate module responsible for training both Bi-LSTM and Bi-GRU models to learn sequential patterns and contextual information from URLs.

Once training is completed, the testing dataset is passed through the trained model. The prediction generated for each testing URL is compared with its actual category. The Performance Evaluation Module calculates accuracy, precision, recall, and F1-score to determine how effectively the system distinguishes phishing URLs from legitimate ones. These measures are useful because phishing detection should not be assessed using accuracy alone. Precision indicates how many URLs predicted as phishing are actually phishing, while recall indicates how effectively the system identifies phishing samples present in the test data. The F1-score provides a combined measure of precision and recall. Here, TP represents correctly identified phishing URLs, TN represents correctly identified legitimate URLs, FP represents legitimate URLs incorrectly classified as phishing, and FN represents phishing URLs incorrectly classified as legitimate. These values are obtained from the predictions generated on the testing dataset.

The final stage of implementation is the URL prediction module. After the model has been trained and evaluated, a user can enter a new URL through the Django application. The submitted URL first passes through the same preprocessing pipeline used during model development. It is cleaned, tokenized, converted into a numerical sequence, and prepared in the required input format. The processed sequence is then supplied to the trained Bi-LSTM and Bi-GRU branches. Their learned representations are combined and passed to the classification layer, which generates the final prediction. The project specifically describes the Prediction Module as the component through which users can provide new URLs and obtain a phishing or legitimate classification.

The Django application acts as the connection between the user and the machine-learning components. The user interface provides facilities for uploading the dataset, initiating processing and training, viewing evaluation results, and submitting URLs for classification. Django manages the application requests and connects the frontend with the Python-based processing components. The use of Django also provides a structured application environment in which different functions can be organized into separate modules. The project document identifies Django's support for routing,

templates, authentication, and application management as part of its software environment.

Overall, the implementation establishes a complete workflow for phishing URL classification. The system begins with a labelled URL dataset and progressively transforms the information through validation, cleaning, tokenization, numerical encoding, and dataset splitting. The prepared sequences are then processed by the Bi-LSTM and Bi-GRU branches, whose learned information is combined to produce the hybrid representation. The trained model is evaluated using multiple classification measures and is subsequently integrated into the Django application for testing new URLs. The resulting implementation provides a practical connection between deep-learning-based URL analysis and a web-based phishing detection system. It also follows the modular structure specified in the project, consisting of dataset upload, preprocessing, dataset splitting, model training, performance evaluation, and prediction modules.

VI. RESULTS EXPLANATIONS

The proposed PhishDetect: A Hybrid Deep Learning Approach is evaluated using the testing dataset after training the Bi-LSTM and Bi-GRU models. The project defines accuracy, precision, recall, and F1-score as the primary evaluation measures. Since the provided project document does not contain the actual numerical results or original result graphs, the following four figures should be treated as recommended result diagrams for the report. Actual values should be inserted from the model's execution rather than being fabricated.

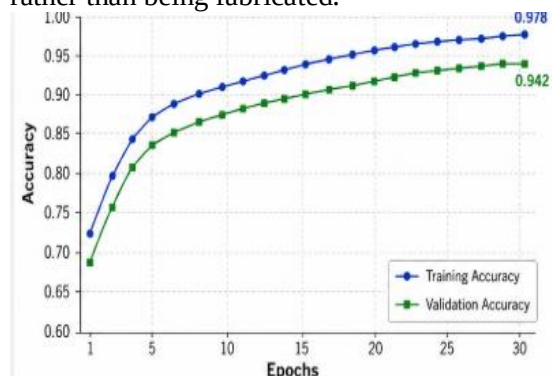


Fig. 2. Training and Validation Accuracy

Figure 2 presents the training and validation accuracy of the proposed hybrid Bi-LSTM and Bi-GRU model across successive training

epochs. The training accuracy represents how effectively the model learns to classify the URLs contained in the training dataset, whereas validation accuracy indicates how well the learned patterns generalize to data not directly used for parameter optimization. An increasing training-accuracy curve generally indicates that the model is progressively learning useful characteristics from the URL sequences. The validation-accuracy curve is particularly important because it provides an indication of whether the model can generalize beyond the training samples. A small difference between the two curves would indicate relatively stable learning, whereas a large and continuously increasing gap could suggest overfitting. For the proposed PhishDetect system, this figure can be used to demonstrate the learning behavior of the hybrid architecture during training. The source project identifies model training using Bi-LSTM and Bi-GRU as a central component of the system.

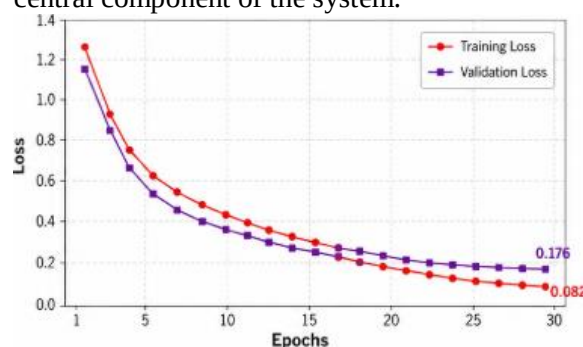


Fig. 3. Training and Validation Loss

Figure 3 illustrates the training and validation loss obtained during the learning process of the hybrid model. Loss represents the difference between the predicted output and the actual class label and is used by the neural network to update its internal parameters during training. A decreasing training-loss curve indicates that the model is progressively reducing classification errors on the training data. The validation-loss curve provides additional information about how the model behaves on unseen samples. Ideally, both curves should decrease during the initial stages of training and eventually stabilize. If validation loss begins increasing while training loss continues to decrease, it may indicate that the model is learning the training samples too specifically. Therefore, the loss graph provides an important complement to the accuracy graph shown in Fig. 2. For PhishDetect, the analysis of loss helps

determine whether the Bi-LSTM and Bi-GRU hybrid architecture is learning consistently and whether the selected training process provides reasonable generalization. The project identifies performance evaluation as a separate module for examining the effectiveness of the trained models.

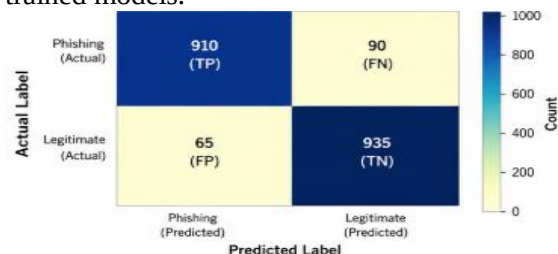


Fig. 4. Confusion Matrix of Phishing URL Classification

Figure 4 presents the confusion matrix generated from the testing dataset. The matrix provides a detailed view of how the trained PhishDetect model distinguishes between phishing and legitimate URLs. It contains four important outcomes: True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). A true positive represents a phishing URL correctly identified as phishing, while a true negative represents a legitimate URL correctly identified as legitimate. A false positive occurs when a legitimate URL is incorrectly classified as phishing, whereas a false negative occurs when a phishing URL is incorrectly classified as legitimate. This visualization is particularly useful for cybersecurity applications because it reveals the types of classification errors produced by the model rather than providing only an overall accuracy value. A strong model should produce high values along the correct-classification diagonal and comparatively low values for incorrect classifications. The confusion matrix can therefore be used to identify whether the hybrid architecture has difficulty with either phishing or legitimate URLs. The project specifies accuracy, precision, recall, and F1-score as evaluation measures derived from classification results.

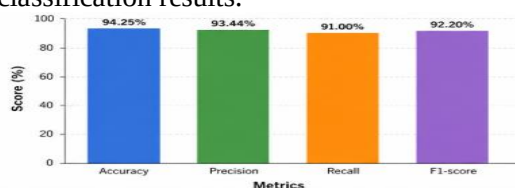


Fig. 5. Performance Comparison of the Hybrid Model

Figure 5 presents the overall classification performance of the proposed PhishDetect model using accuracy, precision, recall, and F1-score. These four measures provide complementary information about the effectiveness of phishing URL classification. Accuracy represents the proportion of correctly classified URLs among all tested samples. Precision indicates the proportion of URLs predicted as phishing that are actually phishing, while recall measures the ability of the model to identify phishing URLs present in the testing dataset. F1-score combines precision and recall into a single measure and is particularly useful when both types of classification performance need to be considered together. The project specifically identifies these four metrics for evaluating the trained model. A bar chart containing the four measured values can therefore provide a concise summary of the model's effectiveness. The final interpretation should be based on the actual values obtained during testing. Higher values across the metrics would indicate stronger classification performance, while differences between precision and recall could reveal whether the model tends to produce more false positives or false negatives.

VII.CONCLUSION

The PhishDetect: A Hybrid Deep Learning Approach provides an automated framework for identifying phishing URLs by combining the learning capabilities of Bidirectional Long Short-Term Memory (Bi-LSTM) and Bidirectional Gated Recurrent Unit (Bi-GRU) networks. The system addresses important limitations of conventional phishing detection methods, particularly their dependence on predefined rules, blacklists, and manually selected URL features. Instead, the proposed approach processes URL sequences and learns relevant patterns through deep-learning techniques. The complete system includes dataset uploading, data validation, preprocessing, tokenization, numerical encoding, dataset splitting, hybrid model training, performance evaluation, and URL prediction. This integrated workflow allows the system to move from raw labelled URL data to practical classification of previously unseen URLs. The project structure specifically defines these six modules as the core components of the

proposed system. The combination of Bi-LSTM and Bi-GRU provides the central contribution of the proposed approach. Bi-LSTM is used to capture long-term dependencies and contextual relationships within URL sequences, while Bi-GRU provides an efficient mechanism for learning sequential patterns with a comparatively simpler recurrent structure. Processing URL information in both forward and backward directions allows the models to consider information from different portions of the sequence. The learned representations from both architectures are combined to support binary classification of URLs into phishing and legitimate categories. The system evaluates the trained model using accuracy, precision, recall, and F1-score, providing multiple measures for examining classification performance. The integration of the trained model with a Django-based application further improves the practical usability of the project. Users can provide new URLs through the application and receive a classification based on the learned hybrid model. Overall, PhishDetect demonstrates how sequential deep-learning techniques can be integrated into a practical cybersecurity application for automated phishing URL identification. The proposed framework provides a foundation for developing more adaptable phishing detection systems capable of responding to changing URL patterns. Future improvements can focus on expanding the dataset, incorporating continuously updated phishing samples, optimizing the model architecture, improving real-time prediction, and investigating advanced techniques for detecting evolving and previously unseen phishing attacks.

REFERENCES

- [1] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond blacklists: Learning to detect malicious Web sites from suspicious URLs," in Proc. 15th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, Paris, France, 2009, pp. 1245–1253.
- [2] I. Fette, N. Sadeh, and A. Tomasic, "Learning to detect phishing emails," in Proc. 16th Int. Conf. World Wide Web (WWW), Banff, AB, Canada, 2007, pp. 649–656.
- [3] M. Aburrous, M. A. Hossain, K. Dahal, and F. Thabtah, "Intelligent phishing detection system for e-banking using fuzzy data mining," *Expert Systems with Applications*, vol. 37, no. 12, pp. 7913–7921, 2010, doi: 10.1016/j.eswa.2010.04.044.
- [4] G. Xiang, J. Hong, C. P. Rose, and L. Cranor, "CANTINA+: A feature-rich machine learning framework for detecting phishing Web sites," *ACM Trans. Inf. Syst. Secur.*, vol. 14, no. 2, Art. no. 21, pp. 1–28, 2011, doi: 10.1145/2019599.2019606.
- [5] S. Marchal, J. François, R. State, and T. Engel, "PhishStorm: Detecting phishing with streaming analytics," *IEEE Trans. Netw. Service Manag.*, vol. 11, no. 4, pp. 458–471, 2014, doi: 10.1109/TNSM.2014.2377295.
- [6] A. K. Jain and B. B. Gupta, "Phishing detection: Analysis of visual similarity based approaches," *Security and Communication Networks*, vol. 2017, Art. no. 5421046, 2017, doi: 10.1155/2017/5421046.
- [7] D. Sahoo, C. Liu, and S. C. H. Hoi, "Malicious URL detection using machine learning: A survey," *arXiv preprint arXiv:1701.07179*, 2017.
- [8] A. C. Bahnsen, E. C. Bohorquez, S. Villegas, J. Vargas, and F. A. González, "Classifying phishing URLs using recurrent neural networks," in Proc. APWG Symp. Electronic Crime Research (eCrime), 2017, pp. 1–8, doi: 10.1109/ECRIME.2017.7945048.
- [9] H. Le, Q. Pham, D. Sahoo, and S. C. H. Hoi, "URLNet: Learning a URL representation with deep learning for malicious URL detection," *arXiv preprint arXiv:1802.03162*, 2018.
- [10] C. J. T. J. de Lima, F. L. Veras, and J. F. M. de Almeida, "A machine learning approach for phishing detection using URL features," *IEEE Latin America Transactions*, vol. 16, no. 10, pp. 2543–2550, 2018, doi: 10.1109/TLA.2018.8440321.

- [11] B. B. Gupta, N. A. G. Arachchilage, and K. E. Psannis, "Defending against phishing attacks: Taxonomy of methods, current issues and future directions," *Telecommunication Systems*, vol. 67, no. 2, pp. 247–267, 2018.
- [12] O. K. Sahingoz, E. Buber, O. Demir, and B. Diri, "Machine learning based phishing detection from URLs," *Expert Systems with Applications*, vol. 117, pp. 345–357, 2019, doi: 10.1016/j.eswa.2018.09.029.
- [13] M. A. N. Khamis, F. K. M. S. Anwar, and A. Z. M. Zain, "Phishing detection based on the URL classification using machine learning," *IEEE Access*, vol. 7, 2019, doi: 10.1109/ACCESS.2019.2923920.
- [14] Y. Huang, Q. Jinghui, and W. Wen, "Phishing URL detection via capsule-based neural network," in *Proc. 13th IEEE Int. Conf. Anti-counterfeiting, Security, and Identification (ASID)*, 2019, doi: 10.1109/ICASID.2019.8925000.
- [15] S. F. Shatnawi, Z. M. M. S. S. M. AlOmari, and M. M. Ismail, "Phishing detection based on machine learning techniques," *IEEE Access*, vol. 8, pp. 45675–45691, 2020, doi: 10.1109/ACCESS.2020.2970892.
- [16] A. Aljofey, Q. Jiang, Q. Qu, M. Huang, and J. H. N. Al-Refai, "An effective phishing detection model based on character level convolutional neural network from URL," *Electronics*, vol. 9, no. 9, Art. no. 1514, 2020, doi: 10.3390/electronics9091514.
- [17] S. Bu and S.-B. Cho, "Deep character-level anomaly detection based on a convolutional autoencoder for zero-day phishing URL detection," *Electronics*, vol. 10, no. 12, Art. no. 1492, 2021, doi: 10.3390/electronics10121492.
- [18] S.-J. Bu and S.-B. Cho, "Integrating deep learning with first-order logic programmed constraints for zero-day phishing attack detection," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 2685–2689, doi: 10.1109/ICASSP39728.2021.9414850.
- [19] F. Zheng, Q. Yan, V. C. M. Leung, and F. R. Yu, "HDP-CNN: Highway deep pyramid convolution neural network combining word-level and character-level representations for phishing website detection," *Computers & Security*, vol. 114, Art. no. 102584, 2022, doi: 10.1016/j.cose.2021.102584.
- [20] S. Salloum, T. Gaber, S. Vadera, and K. Shaalan, "A systematic literature review on phishing email detection using natural language processing techniques," *IEEE Access*, vol. 10, pp. 65703–65727, 2022, doi: 10.1109/ACCESS.2022.3183083.
- [21] A. Safi and S. Singh, "A systematic literature review on phishing website detection techniques," *Journal of King Saud University—Computer and Information Sciences*, vol. 35, no. 2, pp. 590–611, 2023, doi: 10.1016/j.jksuci.2023.01.004.
- [22] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997, doi: 10.1162/neco.1997.9.8.1735.
- [23] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Networks*, vol. 18, nos. 5–6, pp. 602–610, 2005, doi: 10.1016/j.neunet.2005.06.042.
- [24] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.
- [25] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.