Journal of ICT, Design, Engineering and Technological Science (JITDETS)
VOL. 9, NO. 2, pp. 105-116, 2025
DOI: https://doi.org/10.33150/JITDETS-9.2.5

ORIGINAL CONTRIBUTION

A Hybrid NLP and Deep Learning Framework for Phishing Detection in Emails and URLs

Basheer Riskhan[1*], Md Saiful Arefin[2], Mutasim Billah[3], Abdullah Al Hadi[4], Siti Shafrah Shahawai[5], Siva Raja Sindiramutty[6], Noor Zaman Jhanjhi[7]

[1,2,3,4,5]School of Computing and Informatics, Albukhary International University, Kedah, Malaysia

[6,7]School of Computer Science, Taylor's University, Malaysia

***Abstract*— Phishing attacks are constantly evolving, exploiting users with malicious URLs and misleading emails, while conventional rule-based detection methods struggle to keep pace with new threats. To improve detection accuracy and adaptability, this study proposes a hybrid phishing detection framework that combines Deep Learning (DL) and Natural Language Processing (NLP) techniques. For email classification, the system uses TF-IDF-based feature extraction, including word- and character-level n-grams, domain encoding, and link-count analysis; for URL analysis, character-level tokenisation and manually created structural features are used. In addition to CNN, LSTM, and Hybrid CNN-LSTM models for URL classification, three deep learning architectures are developed for email detection: Convolutional Neural Network (CNN), Bidirectional Long Short-Term Memory (BiLSTM), and a Hybrid CNN-BiLSTM model. The hybrid architectures efficiently capture intricate phishing patterns by combining sequential dependency learning with spatial feature extraction. Both primary email and large-scale URL datasets are used, with stratified data partitioning and suitable preprocessing methods, to assess the proposed framework. The methodology addresses the drawbacks of static, single-model systems in contemporary cybersecurity environments by demonstrating a scalable, flexible approach to phishing detection.**

***Index Terms*— Phishing Detection, NLP; Deep Learning, CNN-BiLSTM, TF-IDF, Email and URL Classification, Hybrid Model, Cybersecurity**

Received: 7 August 2025; **Accepted:** 13 October 2025; **Published:** 19 December 2025

I. INTRODUCTION

In the modern era, online phishing attacks pose a significant risk, as they target individuals and organisations to steal confidential data [1]. Conventional phishing defences use heuristics to identify phishing attempts by looking at email headers and URL patterns. Basic phishing attempts are detected by standard phishing detection systems, but these systems are unable to recognise novel approaches and the sophisticated evasive strategies employed by cybercriminals. Online hackers use social manipulation to trick users and alter email content to conceal their methods, making it impossible for basic detection systems to stop them. Because users are no longer protected from evolving phishing attacks by current defence systems, modern detection techniques must advance. New developments in NLP and DL technologies show promise for improving phishing threat detection [2]. NLP tools use both writing structure and context understanding to analyse email wording and identify phishing patterns [3]. Certain warning indicators, such as misleading language, urgent requests, and dubious links, are common in phishing emails and can help security systems identify threats. Because NLP-based models can handle new types of data and defend against novel strategies, they can identify phishing attempts more accurately [4, 5].

When deep learning models like CNNs and BiLSTMs extract intricate features from massive datasets, phishing detection improves [6, 7]. While CNNs identify patterns in the structure of email content, BiLSTMs examine the timing of email content to see how it evolves over messages. GCNs are a novel approach to phishing detection that outperforms traditional techniques by analysing word connections in email texts [8, 9]. Phishing detection is still struggling to achieve optimal results. A key challenge is these systems' ability to keep pace with emerging attacks. Nonetheless, the combination of NLP and DL has the potential to deliver superior solutions. Such solutions will enhance the identification of new threats and reduce false positives [10]. Future studies should aim to strengthen these models to enhance their practical application.

A. Problem statement

Phishing attacks are dynamic, posing a significant challenge for developers who struggle to keep pace with evolving security measures. One challenge in detecting phishing is developing a system that is both flexible and robust. Attackers regularly change email content, sender details, and URLs to bypass fixed detection models [11, 12]). Simple detection mechanisms often fail to deal with advanced phishing schemes because they rely on fixed patterns. Modern phishing detectors need dynamic capabilities, as [8] point out, to address new types of attacks. It is becoming increasingly challenging to develop effective phishing detection because there are limited high-quality training and testing materials. Machine learning and deep learn-

*Corresponding author: Basheer Riskhan
†Email: b.riskhan@aiu.edu.my

ing models are trained on high-quality datasets to detect phishing attacks across different platforms ([3, 13], yet obtaining precisely labeled phishing examples remains a major challenge. A small set of established phishing techniques can constrain a model's ability to learn new ones, leading to spurious security warnings. Moreover, phishing defense systems often flag legitimate websites as malicious, which frustrates users and erodes their trust in the security system, leading them to ignore trusted warnings [8, 14]. It is a problem especially encountered by organizations that use automated mechanisms to detect phishing attempts when securing sensitive information. A good phishing detection system must be highly engineered, with robust feature engineering and model optimization, to reduce false positives and ensure a high detection rate.

Deep learning models pose a significant challenge due to their high computational demands. For example, real-time email analyzers must process large volumes of email quickly [15]. The high computation requirements of these algorithms render it hard to implement deep learning techniques in resource-limited settings. We are primarily focused on developing a more effective phishing detection system that can adapt to evolving threats, handle data efficiently, and be user-friendly. The subsequent research must focus on innovative approaches to adjust detection systems, the effective availability of robust training data, and the speed of performance to develop practical and effective schemes for phishing protection [3].

B. Research objectives

The key objectives of this study are three-fold: first, analyze existing studies and techniques in phishing detection, namely, how NLP and DL are applied, to identify aspects that require improvement and lack clear solutions. Second, to create an innovative hybrid model that combines NLP to examine content in depth and DL to automatically identify features, resulting in a superior, more efficient system for phishing detection. Third, to determine whether the proposed model is effective based on original and existing data, it is necessary to focus on metrics such as accuracy, precision, recall, and false-positive rate.

C. Contributions

This work establishes a research direction while offering basic ideas for improving cybersecurity by developing a hybrid phishing-detection system. The research outcomes can help expert teams design better phishing-detection systems and find new ways to work and to fix existing system issues. The research supports the development of AI security systems by its efforts to stay ahead of phishing defenses.

The Hybrid CNN-BiLSTM combined the convolutional and sequential processing. The input sequence was embedded, and then processed simultaneously using two stream sections: one section passed through a Conv1D layer with 128 neurons and a GlobalMaxPooling1D, and the other section passed through a Bidirectional LSTM layer with 64 units. The outputs of the two branches were stacked and fed to a dense layer of 64 ReLU units, a dropout layer with a 0.5 rate, and a final output layer with sigmoid activation. This model also used the Adam optimizer and binary cross-entropy loss, and performance was measured by accuracy and AUC.

II. RELATED WORK

A. Traditional phishing detection

1) Blacklist checking

Security systems built on blacklists block URLs from lists maintained by Google Safe Browsing, PhishTank, and the APWG. The organizations keep these lists up to date as part of their efforts to block access to phishing sites. This method lets you deploy defense systems easily because it keeps things simple. Users either find their access to blacklisted URLs blocked or receive a notification when trying to access them. Blacklisting fails to safeguard users against attacks that have emerged since the last database update, despite being used as a preventive measure. The blacklists cannot be effective because [16, 17] show that attackers can modify their URLs and web page content with only limited details. It is not easy to maintain an up-to-date, exhaustive blacklist of phishing sites, given the continuously evolving nature of such attacks [18, 19].

2) Heuristic-based approaches

Phishing detection systems rely on heuristics and patterns, using predefined rules to identify potential phishing attempts in URLs, web pages, and emails. They examine digital clues such as multiple subdomains pointing to the same URL, grammatical mistakes, extremely long addresses, domain names substituted for IP addresses, and keywords that indicate phishing. They also analyze HTML text and metadata to identify covert form fields, obfuscated script code, and excessive redirects [20, 21]. They can identify phishing sites using URL pattern analysis without relying on blocking lists. A typical disadvantage, however, is the tendency to classify good websites as malicious, since phishing and legitimate websites often look very similar. Moreover, heuristic rules need regular updates to keep up with changes in the techniques attackers use [18].

B. Machine learning-based detection

Machine learning can improve the ability to detect phishing attacks by enabling the development of more effective detection techniques [22]. Machine learning techniques have been used to analyze a wider range of URL and site features, as well as network traffic, to better detect phishing. [16] indicate that this detection system brings together three effective machine learning strategies, such as supervised, unsupervised, and reinforcement learning, to enhance the level of accuracy in phishing detection. One of the major benefits of machine learning in this scenario is its ability to detect new attacks by analyzing patterns in data from large datasets. In particular, phishing attacks are detected using specialized textual sequences within URLs, along with WHOIS registration information and web page layout characteristics [18, 23]. According to the study by [20], the Decision Tree, Random Forest, Support Vector Machine, CNN, and BERT models are effective in detecting phishing attacks. ML-based solutions deliver strong performance but encounter practical difficulties. We face a crucial barrier to obtaining sufficiently large, up-to-date data sets needed to build effective models. Specific ML systems remain vulnerable to defeat when attackers intentionally manipulate website elements to hide their actions. The heavy processing requirements of advanced ML models make them ineffective for networks with low computational capacity, according to [16, 24]

C. NLP in phishing detection

Natural Language Processing serves as a key cybersecurity solution because it automatically analyzes and makes sense of content users create [25]. NLP analyzes vast amounts of text data to detect security threats, including phishing emails, social engineering attacks, and fake news. NLP analyzes text in different communication platforms to spot signs of cyber threats [26, 27]

1) Understanding and processing user-generated content for security purposes

NLP detects security threats in user content by analyzing social media posts and customer feedback using its methods [28]. We use methods such

as tokenization, named entity recognition, and sentiment analysis to identify key details and categorize text as safe or dangerous. NLP technology detects phishing attempts through these methods by recognizing signs of social engineering attacks in writing. Security systems use NLP to analyze the context of phishing emails to more accurately distinguish genuine from dangerous content [8, 29].

2) NLP techniques relevant to phishing detection

NLP techniques for detecting phishing show strong results in sentiment analysis, text classification, and topic modeling [30]. Support vector machines and deep learning models work together with text classification to find phishing content in emails, while sentiment analysis looks for dangerous emotions in messages. The TF-IDF and word embedding methods help improve the performance of phishing detection systems by recognizing specific patterns in these emails, according to [31]. Cybersecurity systems gain better protection against phishing attacks through NLP methods such as lsi. Task response varies based on email content.

D. Deep learning architectures

Deep learning technology helps cybersecurity professionals detect threats by automatically finding patterns and identifying unusual activities. New deep learning techniques deliver effective tools to spot phishing attempts, block malware, and prevent unauthorized access to systems [32, 33].

1) Recent advancements in deep learning applicable to cybersecurity

Researchers now use advanced DL techniques, including transformer models with attention mechanisms, and extend them through hybrid approaches using multiple neural network architectures to boost detection results. BiLSTM and CNNs work together to detect phishing attacks effectively by jointly extracting time-based and location-based features, as reported by [34]. According to [35], pre-trained models and transfer learning enable cybersecurity systems to rapidly adapt to and combat emerging threats, significantly reducing the need for extensive retraining.

2) Deep learning architectures for phishing detection

The deep learning frameworks, CNNs and RNNs, demonstrate a strong ability to identify phishing threats. CNNs detect design features in phishing web pages, while RNNs with LSTM and GRU units effectively analyze email content and URLs, according to [36]. Modified deep neural networks with LSTM functionality, which enable them to retain longer-term short-term memory, are useful for tracking phishing URLs based on local links and long-distance dependency associations [34]. Deep learning demonstrates how it could revolutionize cybersecurity by identifying cyber threats as they occur and evolving to new patterns.

E. Research gaps

1) Limitations of existing phishing detection systems

Even though there have been recent improvements in phishing detection systems, technical constraints remain. The main problem for researchers is the prevalence of false positives: legitimate websites are mistaken for phishing sites, leading to a negative user experience [37]. Moreover, existing detection strategies are unlikely to be effective at detecting new attacks, as they rely on past data and familiar patterns [38]. The limited availability of current phishing datasets makes it hard for detection models to reliably identify different attack types [39]. Running deep learning models on mobile devices is difficult due to their limited power and battery resources [40]. Current detection methods do not perform well on phishing content written in different languages, as most training data consists only of English texts.

2) Addressing challenges with NLP and Deep learning

The latest advances in phishing-detection systems have failed to overcome their fundamental technical problems. Researchers identify too many legitimate websites as phishing sites, which harms the user experience when dealing with real websites. The current detection system fails to detect new attacks as it relies on previous incident records and standardized patterns to perform its tasks [38]. Detecting different attack types becomes challenging for detection models due to insufficient access to modern phishing datasets. [39] Mobile devices have insufficient CPU power and battery capacity, making the execution of deep learning models challenging. [40]. Most phishing detection systems struggle to identify foreign-language phishing content because their training data consists only of English texts.

III. SUMMARY OF FINDINGS

The analysis reveals that researchers have developed phishing detection from basic blacklists and heuristic rules into complex machine learning and deep learning solutions. Existing blacklist detection tools work well against established phishing attacks because they use identified threats in database files. The flexibility of heuristic detection methods leads to many incorrect results, undermining their trustworthiness. Machine learning now detects phishing with high accuracy by leveraging information from URLs, website content, and metadata using modern machine learning methods. SVMs and decision trees perform well as supervised learning models, but deep learning methods such as CNNs and transformer models achieve superior results at finding complex data patterns. Phishing detection technology is advancing rapidly, but faces challenges such as a lack of high-quality data, significant processing requirements, and vulnerabilities to cyber-attacks. Natural Language Processing (NLP) offers a solution by analyzing text in emails and web pages to pinpoint phishing indicators. Research shows that text analysis approaches such as sentiment analysis, topic modeling, and text classification can detect phishing better than before. Deep learning tools now include new hybrid systems combining CNNs and recurrent networks to better detect phishing activity. Table 1 summarizes the findings of related work.

TABLE I
RESEARCH ANALYSIS

Title of the Paper	Author and Year	Method	Research Gap
A novel hybrid approach of SVM combined with NLP and probabilistic neural network for email phishing	[31]	In this research study, SVM, NLP, and PNN are used for phishing detection. It uses message and tokenization to extract features, deletes stop words and performs a rigid and elastic classification for achieving high accuracy on 1705 inbox data.	The current approaches have no provision for dynamic adaptation and the presence of multiple classifiers. To overcome this, the paper brings out the hybrid SVM-PNN with enhanced feature selection and better accuracy.
Intelligent Deep Machine Learning Cyber Phishing URL Detection Based on BERT Features Extraction	[37]	For feature extraction, this paper employs BERT and as a phishing URL detection technique, a deep learning technique known as Convolutional Neural Networks (CNNs) is applied. Featuring NLP to the content, the dataset contains 472,259 entries. It is integrated into a recommendation list, which has an accuracy of 96.66%.	Since they are not very effective in feature extraction, conventional techniques fail in the accurate identification of the phishing URLs because of their complexities. This kind of research question is solved by this study with the help of BERT with respect to classic ML for the effective feature extraction.
Detection of Phishing Websites by Using Machine Learning-Based URL Analysis	[20]	Machine learning algorithms analyze URL features to detect phishing without relying on third-party services.	Existing methods lack real-time detection and adaptability to evolving phishing tactics.
Email phishing: Text classification using natural language processing	[41]	Preprocessing of text data in the study involves tokenization, stemming, and stop-word removal from the textual data. Other common classifiers such as Support Vector Machine (SVM), Decision Trees, Random Forest, and Naive Bayes are used in this study to categorize phishing emails. These messages (ham/spam) contain 5574 messages, successfully getting an accuracy up to 98.77% by applying SVM.	Previous approaches to detection had poor preprocessing and time consuming classification for phishing mails. This study redresses this default by adopting NLP and multiple ML classifiers to increase the accuracy and to accommodate such a rich structured and unstructured text data.
Phishing Website Detection through Multi-Model Analysis of HTML Content	[16]	A fusion model combining MLP and NLP models (CANINE and RoBERTa) to analyze HTML content for phishing detection. The approach achieves high accuracy and F1-score through feature extraction and model fusion.	Limited availability of recent datasets and lack of exclusive focus on HTML content in existing studies. The study addresses this by creating a new dataset and introducing a novel model fusion technique.
An In-Depth Benchmarking and Evaluation of Phishing Detection Research for Security Needs	[18]	Introduces 'PhishBench,' a benchmarking framework to evaluate phishing detection methods using a unified system and diverse datasets.	Existing methods lack standardized evaluation frameworks, leading to inconsistencies in assessing phishing detection techniques.

IV. PROPOSED METHODOLOGY

A. System overview

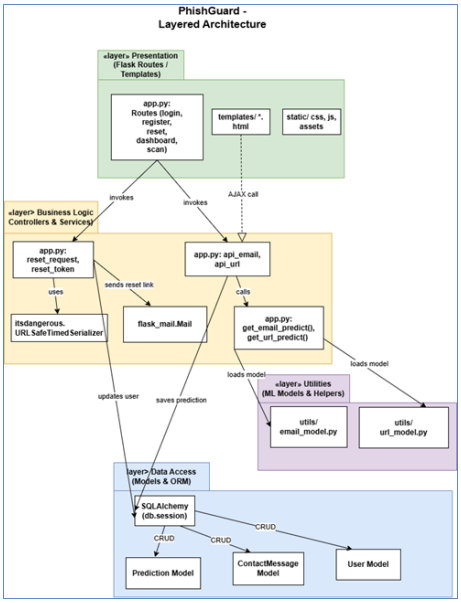

Fig. 1 *Layered architecture*

Figure 1 shows a layered architecture diagram that involves a presentation application and a data layer. The PhishGuard system follows a three-layered architecture, which logically separates responsibilities into:

1) Presentation layer

The system PhishGuard Presentation Layer is the main interface through which users communicate with the application. It was built on the Flask web framework, using Jinja2 templating, HTML, CSS, JavaScript, and Bootstrap 5 to provide a modern, responsive, and user-friendly experience. This layer is responsible for displaying all components on the client side, receiving user inputs, and maintaining smooth communication with the backend.

There are a few important pages on the interface: login (/login), registration (/register), email scanner (/email), URL scanner (/url), dashboard (/dashboard), and contact form (/contact). All pages are coded in HTML for structural layout, designed with CSS and Bootstrap 5 for visual appearance and device responsiveness, and refined with JavaScript and AJAX to support dynamic content updates. For example, when a user sends an email file or a URL link to be evaluated, the data will be sent via AJAX as an asynchronous parameter to the server-side endpoints (/api/email or /api/url). After processing, the backend returns a prediction, which is then displayed dynamically on the page without reloading the entire page. In general, the interface should be usable and friendly to users with varying levels of technical expertise. Real-time hint messages, user-friendliness, layouts, and navigation are features that will enhance navigation and viewer involvement. In addition, security is one of the factors in the presentation layer. The Jinja2 autoescape feature helps reduce the

number of cross-site scripting (XSS) exploits. Session management and optional CSRF libraries are used to prevent unauthorized user actions.

In a nutshell, the Presentation Layer serves as a gateway for connecting users to PhishGuard's smart phishing-detection services. It can not only ensure efficient communication between users but also maintain the level of a secure, responsive real-time threat assessment platform.

2) Application layer (Business logic)

The Application Layer is the heart of the operational system, PhishGuard, performing user interaction, implementing system logic, and maintaining communication between the server interface and the database. The layer was created with the Flask web framework in Python and wraps the business logic needed for authentication, execution of phishing predictions, user data management, and API endpoints with dynamic actions.

The layer integrates several crucial frameworks and libraries to achieve durability and division. Flask provides the foundation for routing, receiving, and processing HTTP requests and responses, as well as for designing the overall application flow. Session management and user authentication are handled with Flask-Login; secure password reset features use itsdangerous to generate tokens, and Flask-Mail to send emails. It uses SQLAlchemy as the Object-Relational Mapping (ORM) instrument to simplify communication between the database and the application. All important routes and API endpoints are determined by the Application Layer. They include login, registration, email verification, web address scanning, dashboard visualization, and contact inquiry forms. Access control is enforced using self-defined, login-required decorators, ensuring that only authorized users can access restricted resources. In addition, the system includes the full password reset procedure, which combines tokens, password reset links, password hashing, and email reminders to the user.

The two important API endpoints are where the .eml files are uploaded and where the raw URLs are sent, respectively, and are accessed via api/email and api/url. Article Rewriter can be used to enhance article writing without plagiarism. It works with content generated by deep learning models (lazy-loaded via the get_email_predict and get_url_predict functions). These models have built-in utility modules for pre-processing and prediction. The scan results, along with metadata such as prediction confidence, type, and creation date, are stored in a database to support dashboard analytics and user history. In addition to prediction workflows, the Application Layer provides endpoints at/api/history and /api/stats to give users and administrators access to historical data and real-time system statistics. The data validation process is being strictly followed to eliminate incomplete or invalid files and URLs. Data confidentiality is ensured by not storing unprocessed email messages or other user-sensitive data. The content domain's standard is Standard.

The Application Layer serves as the brain of the PhishGuard platform. It leverages a set of formidable technologies, including strong authentication, data validation, predictive modeling, and secure logging practices. Its scalable and secure architecture not only guarantees scalability but also lays the groundwork for future improvements to the system.

3) Data layer (Database and ORM integration)

All persistent storage is done by the Data Layer of the PhishGuard system. It will ensure the secure and efficient storage of sensitive data, such as user credentials, phishing prediction results, and contact messages. This layer uses the SQLite file-based relational database engine, which is small and ideal for small-scale and local deployments. We also use SQLAlchemy, a Python Object-Relational Mapper (ORM), to provide a smooth interaction with databases.

SQLite was chosen because it was easy to set up, simple, and easy to port, and was mainly needed for development and demonstration. The phishguard.db database file is located in the system instance/- directory, making it easy to access and manipulate the version. SQLAlchemy reduces the complexity of connecting both Python objects and database tables, making manual use of SQL queries unnecessary while providing concise and robust transaction, schema, and relationship handling. The system's data schema defines three major models. The User model stores the following information for user authentication: a unique identifier, email address, an encrypted password (based on werkzeug.security), and a flag indicating an administrative role (is_admin). At the Prediction model level, we record every detection of a phishing attempt (user ID as a foreign key to the User table, type of input (email or URL), final prediction verdict, confidence score, and a timestamp). Such a model allows monitoring and evaluating user activity and model performance over the long term. The third model, contact message, records messages submitted through the contact form, the distiller's name, the distiller's email address, the message body, and the time the message was submitted, enabling user inquiries to be received and responded to accordingly.

The Data Layer implements referential integrity via foreign key constraints, especially between the User and Prediction entities, ensuring that stored data remains consistent and reliable. Moreover, sensitive data (passwords) is never stored in plain text; instead, it is first hashed and then stored to ensure the authentication mechanism is more secure and reliable. The bottom line is that the Data Layer provides an appropriate environment to mitigate security risks and scale all data-driven activities across the PhishGuard system. It ensures that user data, system logs, and even interactions are stored forever and remain readily available, as the data is easy to analyze in real time and to audit later.

B. Dataset description

1) Email dataset

The Email dataset is the primary dataset, and the URL dataset is the secondary dataset. In this data set, we have two labels for email messages: SPAM and NOT SPAM. It utilized 1,569 samples following preprocessing.

TABLE II
EMAIL DATASETS OVERVIEW

Dataset	Total Records	Spam/Phishing (Count / %)	Legitimate/Not Spam (Count / %)	Extracted Features
Email	1,569	458 / 29.18%	1,111 / 70.82%	Cleaned body text (TF-IDF word + char n-grams), domain (OHE), link count (scaled), padded sequences

2) URL dataset

URL dataset (Secondary dataset): This dataset comprises URLs labeled as either "Phishing" or "Legitimate". A total of 1,186,756 samples were

used. The following preprocessing steps were applied: Label Mapping and Filtering:

- Converted the type column to lowercase and stripped whitespace.
- Mapped legitimate → 0, and phishing → 1.
- Removed rows with unknown labels.
- Dropped URLs longer than 200 characters.

TABLE III
URL DATASETS OVERVIEW

Dataset	Total Records	Spam/Phishing (Count / %)	Legitimate/Not Spam (Count / %)	Extracted Features
URL	1,186,756	195,128 / 16.45%	991,628 / 83.55%	Cleaned char-level token sequences (maxlen=200), 7 handcrafted features (scaled)

C. Data preprocessing

1) *Email preprocessing*

Preprocessing steps:
Label extraction: regex was used to extract the label, which was located in the feedback
column: Final category: SPAM – 1, Final Classification: NOT SPAM - 0
Text cleaning: Converted text to lowercase. Removed all HTML tags. Replaced embedded URLs with the token URL. Removed all non-alphanumeric characters (kept only letters, numbers, and spaces). Collapsed extra whitespace and stripped leading/trailing spaces.
Duplicate and empty entry removal: Removed duplicate rows based on the cleaned email body.
Removed rows where the cleaned body was empty or only whitespace.
Tokenization and padding: Applied word-level tokenization using Keras Tokenizer with a vocabulary size of 20,000. Converted cleaned text to integer sequences. All sequences were padded or truncated to a fixed length of 500 tokens.

2) *URL preprocessing*

This dataset comprises URLs labeled as either "Phishing" or "Legitimate". A total of 1,186,756 samples were used. The following preprocessing steps were applied: Label Mapping and Filtering: 1. Converted the type column to lowercase and stripped whitespace. 2. Mapped legitimate → 0, and phishing → 1. 3. Removed rows with unknown labels. Dropped URLs longer than 200 characters.

D. Feature engineering

TF-IDF features:
Word-level n-grams: unigram and bigram (max 2,000 features).
Character-level n-grams: 3 to 5 character grams (max 1,000 features).
Domain encoding:
Extracted the recipient domain from the headers field using regex.
One-hot encoded the domain field.
Link count:
Count how many times the token URL appeared in the cleaned body.
Scaled the count using Standard Scaler.
Final features:
Combined all features into a single sparse matrix:
Domain OHE + Link Count + TF-IDF Word + TF-IDF Char
These were used as input features for machine learning models.

E. Model architectures

1) *Email models*

The phishing detection task was approached using separate deep learning models for the Email and URL datasets, tailored to their respective structures and available features.

Email dataset models: The models for classifying emails were designed to process cleaned email bodies, which were converted into token sequences [42]. Three deep learning architectures were implemented and evaluated:

Convolutional Neural Network (CNN): The Convolutional Neural Network (CNN) model was initialized with an embedding layer that transformed the input sequences of length 500 into dense representations with a vocabulary size and embedding dimension. It used a 1D convolutional layer with 128 filters of size 5, followed by a GlobalMaxPooling1D layer to downsample the features. This output was then fed to a dense layer with 64 units and ReLU activation, followed by a dropout layer with a rate of 0.5. The last layer was a dense output layer with a sigmoid activation function for binary classification. The compilation and evaluation were based on the Adam optimizer and the binary cross-entropy loss function, and were measured by accuracy and AUC, respectively. Figure 2 shows the Convolutional Neural Network (CNN) model for email.

```
def build_cnn():
    inp = Input(shape=(MAX_SEQ_LEN,))
    x = Embedding(VOCAB_SIZE, EMBED_DIM, input_length=MAX_SEQ_LEN)(inp)
    x = Conv1D(128, 5, activation='relu')(x)
    x = GlobalMaxPooling1D()(x)
    x = Dense(64, activation='relu')(x)
    x = Dropout(0.5)(x)
    out = Dense(1, activation='sigmoid')(x)
    model = Model(inputs=inp, outputs=out)
    model.compile(
        optimizer='adam',
        loss='binary_crossentropy',
        metrics=['accuracy', tf.keras.metrics.AUC(name='auc')]
    )
    return model
```

Fig. 2 *The Convolutional Neural Network (CNN) model for Email*

Bidirectional Long Short-Term Memory (BiLSTM): The Bidirectional Long Short-Term Memory (BiLSTM) model was also initialized with an embedding layer configured similarly to the CNN model. The injected speech was subsequently passed through a Bidirectional LSTM layer with 64 units to exploit bidirectional dependencies. This was followed by a dense layer with 64 ReLU units, a dropout rate of 0.5, and an output layer with a sigmoid activation for binary classification. Similar to the CNN model, the BiLSTM architecture employed Adam optimizer and binary cross-entropy loss, and accuracy and AUC were used as metrics. Figure 3 shows the Bidirectional Long Short-Term Memory (BiLSTM) model for email.

```python
def build_bilstm():
    inp = Input(shape=(MAX_SEQ_LEN,))
    x = Embedding(VOCAB_SIZE, EMBED_DIM, input_length=MAX_SEQ_LEN)(inp)
    x = Bidirectional(LSTM(64))(x)
    x = Dense(64, activation='relu')(x)
    x = Dropout(0.5)(x)
    out = Dense(1, activation='sigmoid')(x)
    model = Model(inputs=inp, outputs=out)
    model.compile(
        optimizer='adam',
        loss='binary_crossentropy',
        metrics=['accuracy', tf.keras.metrics.AUC(name='auc')]
    )
    return model
```

Fig. 3 *The Bidirectional Long Short-Term Memory (BiLSTM) model for email*

```python
def build_hybrid():
    inp = Input(shape=(MAX_SEQ_LEN,))
    emb = Embedding(VOCAB_SIZE, EMBED_DIM, input_length=MAX_SEQ_LEN)(inp)
    # CNN branch
    c = Conv1D(128, 5, activation='relu')(emb)
    c = GlobalMaxPooling1D()(c)
    # Bi-LSTM branch
    l = Bidirectional(LSTM(64))(emb)
    # Merge branches
    mrg = concatenate([c, l])
    d = Dense(64, activation='relu')(mrg)
    d = Dropout(0.5)(d)
    out = Dense(1, activation='sigmoid')(d)
    model = Model(inputs=inp, outputs=out)
    model.compile(
        optimizer='adam',
        loss='binary_crossentropy',
        metrics=['accuracy', tf.keras.metrics.AUC(name='auc')]
    )
    return model

print("Model-building functions defined: build_cnn(), build_bilstm(), build_hybrid()")
```

Fig. 4 *The Hybrid CNN-BiLSTM model for email*

Hybrid CNN-BiLSTM Architecture: The Hybrid CNN-BiLSTM combined convolutional and sequential processing. Embedding the input sequence was performed and then the sequence was processed simultaneously using two stream sections: one section by passing through a Conv1D layer with 128 neurons and a GlobalMaxPooling1D and the other section by passing through a Bidirectional LSTM layer with 64 units. The results of the two branches were stacked and supplied to the dense layer of 64 ReLU units, dropout layer of 0.5 rate and the final output layer with sigmoid activation. This model also applied Adam optimizer and binary cross entropy loss, and they were measured with accuracy and AUC. Figure 4 shows the Hybrid CNN-Bilstm model for email.

2) URL models

URL dataset omdels: For URL classification, models were developed to handle tokenized character-level URL sequences and handcrafted numeric features extracted from the URL string.

Convolutional Neural Network (CNN) The URL model was made of two types of input, namely sequence and numeric feature input, using the CNN. The sequence input branch began with an embedding layer, followed by a 1D convolutional layer with 128 filters and a kernel size of 5. This was incremented by a Global Max Pooling 1D layer and a dropout layer (rate = 0.5). The numeric feature representation was a vector of hand-designed features, including the URL length, the number of digits, and the number of symbol characters. These two branches were concatenated and fed into a dense layer with 64 ReLU units, dropout of 0.5, and an output layer with a sigmoid activation. The name of this model was CNN_Model. Figure 5 shows the Convolutional Neural Network (CNN) model for URL.

```python
def build_cnn_model(vocab_size: int,
                    max_seq_len: int,
                    numeric_feat_dim: int,
                    embed_dim: int = EMBED_DIM) -> Model:
    # sequence branch
    seq_input = Input(shape=(max_seq_len,), name='seq_input')
    x = Embedding(input_dim=vocab_size,
                  output_dim=embed_dim,
                  input_length=max_seq_len)(seq_input)
    x = Conv1D(128, 5, activation='relu')(x)
    x = GlobalMaxPooling1D()(x)
    x = Dropout(0.5)(x)

    # numeric branch
    num_input = Input(shape=(numeric_feat_dim,), name='numeric_input')

    # fusion + head
    x = concatenate([x, num_input])
    x = Dense(64, activation='relu')(x)
    x = Dropout(0.5)(x)
    output = Dense(1, activation='sigmoid')(x)

    return Model(inputs=[seq_input, num_input], outputs=output, name='CNN_Model')
```

(a)

Model: "CNN_Model"

Layer (type)	Output Shape	Param #	Connected to
seq_input (InputLayer)	(None, 200)	0	-
embedding (Embedding)	(None, 200, 128)	18,176	seq_input[0][0]
conv1d (Conv1D)	(None, 196, 128)	82,048	embedding[0][0]
global_max_pooling1d (GlobalMaxPooling1D)	(None, 128)	0	conv1d[0][0]
dropout (Dropout)	(None, 128)	0	global_max_pooling1d[…
numeric_input (InputLayer)	(None, 7)	0	-
concatenate (Concatenate)	(None, 135)	0	dropout[0][0], numeric_input[0][0]
dense (Dense)	(None, 64)	8,704	concatenate[0][0]
dropout_1 (Dropout)	(None, 64)	0	dense[0][0]
dense_1 (Dense)	(None, 1)	65	dropout_1[0][0]

Total params: 108,993 (425.75 KB)
Trainable params: 108,993 (425.75 KB)
Non-trainable params: 0 (0.00 B)

(b)

Fig. 5. The Convolutional Neural Network (CNN) model for URL

Long Short-Term Memory (LSTM):
The URL LSTM model had a similar construction to the two-input model. The input sequence was filtered through an embedding layer, a Bidirectional LSTM layer with 64 units, and a dropout layer. At the same time, a separate input received the numeric feature vector. The two inputs were concatenated, then fed into a dense layer with 64 ReLU neurons, followed by a dropout layer with a dropout rate of 0.5, and finally a binary classification layer with a sigmoid output. This model was established as LSTM_Model. Figure 6 shows the LSTM-based URL model.

```python
def build_lstm_model(vocab_size: int,
                     max_seq_len: int,
                     numeric_feat_dim: int,
                     embed_dim: int = EMBED_DIM) -> Model:
    seq_input = Input(shape=(max_seq_len,), name='seq_input')
    x = Embedding(input_dim=vocab_size,
                  output_dim=embed_dim,
                  input_length=max_seq_len)(seq_input)
    x = Bidirectional(LSTM(64))(x)
    x = Dropout(0.5)(x)

    num_input = Input(shape=(numeric_feat_dim,), name='numeric_input')

    x = concatenate([x, num_input])
    x = Dense(64, activation='relu')(x)
    x = Dropout(0.5)(x)
    output = Dense(1, activation='sigmoid')(x)

    return Model(inputs=[seq_input, num_input], outputs=output, name='LSTM_Model')
```

(a)

Model: "LSTM_Model"

Layer (type)	Output Shape	Param #	Connected to
seq_input (InputLayer)	(None, 200)	0	-
embedding_1 (Embedding)	(None, 200, 128)	18,176	seq_input[0][0]
bidirectional (Bidirectional)	(None, 128)	98,816	embedding_1[0][0]
dropout_2 (Dropout)	(None, 128)	0	bidirectional[0][0]
numeric_input (InputLayer)	(None, 7)	0	-
concatenate_1 (Concatenate)	(None, 135)	0	dropout_2[0][0], numeric_input[0][0]
dense_2 (Dense)	(None, 64)	8,704	concatenate_1[0][0]
dropout_3 (Dropout)	(None, 64)	0	dense_2[0][0]
dense_3 (Dense)	(None, 1)	65	dropout_3[0][0]

Total params: 125,761 (491.25 KB)
Trainable params: 125,761 (491.25 KB)
Non-trainable params: 0 (0.00 B)

(b)

Fig. 6. The LSTM-based URL model for URL

F. Hybrid CNN-LSTM architecture:

The hybrid CNN-LSTM URL model also employed two branches to process two inputs: tokenized URL sequences and handcrafted numeric features. Both a CNN block and a BiLSTM block used a sequence input. The CNN block used a Conv1D layer (128 filters, kernel size 5), GlobalMaxPooling1D, and dropout. The BiLSTM block contained a Bidirectional LSTM Layer having 64 units and a dropout layer. Outputs of CNN branch, LSTM branch, and target variables were combined and fed to a dense layer with 64 ReLU units, dropout (0.5), and a sigmoid activation as the output layer. Such a model was called Hybrid_CNN_LSTM. Figure 7 shows the Hybrid CNN-LSTM URL model.

```python
def build_hybrid_model(vocab_size: int,
                       max_seq_len: int,
                       numeric_feat_dim: int,
                       embed_dim: int = EMBED_DIM) -> Model:
    # ── sequence input ──────────────────────────────────────
    seq_input = Input(shape=(max_seq_len,), name='seq_input')
    embed = Embedding(input_dim=vocab_size,
                      output_dim=embed_dim,
                      input_length=max_seq_len)(seq_input)

    # CNN branch
    cnn = Conv1D(128, 5, activation='relu')(embed)
    cnn = GlobalMaxPooling1D()(cnn)
    cnn = Dropout(0.5)(cnn)

    # Bi-LSTM branch
    lstm = Bidirectional(LSTM(64))(embed)
    lstm = Dropout(0.5)(lstm)

    # ── numeric features branch ─────────────────────────────
    num_input = Input(shape=(numeric_feat_dim,), name='numeric_input')

    # ── fusion & head ───────────────────────────────────────
    x = concatenate([cnn, lstm, num_input])
    x = Dense(64, activation='relu')(x)
    x = Dropout(0.5)(x)
    output = Dense(1, activation='sigmoid')(x)

    return Model(inputs=[seq_input, num_input], outputs=output,
                 name='Hybrid_CNN_LSTM')
```

(a)

Layer (type)	Output Shape	Param #	Connected to
seq_input (InputLayer)	(None, 200)	0	-
embedding_2 (Embedding)	(None, 200, 128)	18,176	seq_input[0][0]
conv1d_1 (Conv1D)	(None, 196, 128)	82,048	embedding_2[0][0]
global_max_pooling1d_1 (GlobalMaxPooling1D)	(None, 128)	0	conv1d_1[0][0]
bidirectional_1 (Bidirectional)	(None, 128)	98,816	embedding_2[0][0]
dropout_4 (Dropout)	(None, 128)	0	global_max_pooling1d_…
dropout_5 (Dropout)	(None, 128)	0	bidirectional_1[0][0]
numeric_input (InputLayer)	(None, 7)	0	-
concatenate_2 (Concatenate)	(None, 263)	0	dropout_4[0][0], dropout_5[0][0], numeric_input[0][0]
dense_4 (Dense)	(None, 64)	16,896	concatenate_2[0][0]
dropout_6 (Dropout)	(None, 64)	0	dense_4[0][0]
dense_5 (Dense)	(None, 1)	65	dropout_6[0][0]

Total params: 216,001 (843.75 KB)
Trainable params: 216,001 (843.75 KB)
Non-trainable params: 0 (0.00 B)

(b)

Fig. 7. The Hybrid CNN-LSTM URL model

G. Experimental setup

To evaluate the hybrid models, the datasets were partitioned strategically into training, validation, and testing subsets.

Email dataset partitioning: The email dataset was split into a Training Set (80%). Validation Set: 20% was used for validation. Class labels (SPAM = 1, NOT SPAM = 0) were extracted from the feedback column. Features used include TF-IDF vectors (word and character), domain encoding, and link count.

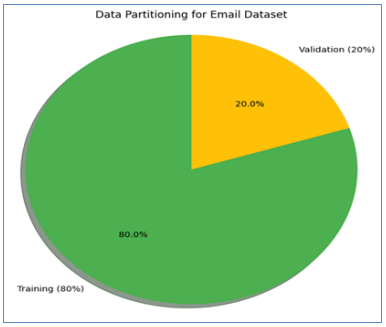

Fig. 8 *Data Partitioning for Email Dataset*

URL Dataset Partitioning:The URL dataset was split into three subsets using stratified sampling:

- Training Set: 70% of the data.
- Validation Set: 15% of the data.
- Test Set: 15% of the data.

The label column was created by mapping phishing to 1 and legitimate to 0. URLs longer than 200 characters were removed before splitting. Random oversampling was applied to the training set to balance the proportions of phishing and legitimate URLs. The validation and test sets remained unchanged and reflected the original class distribution.

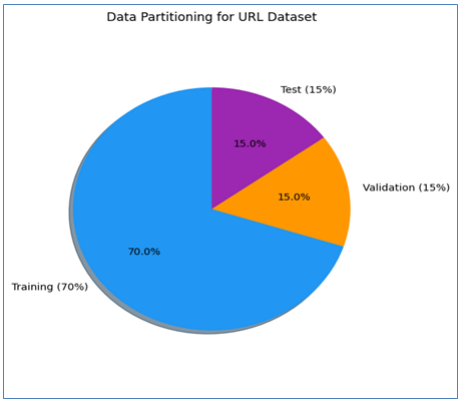

Fig. 9 *Data Partitioning for URL Dataset*

H. System implementation

Incremental development model: The phishing detection system was developed using the Incremental Development Model. This approach facilitated partitioning the project into smaller, manageable modules, allowing for modular implementation and a gradual integration process. First, the system architecture was structured in such a way that the layers are well-separated (presentation layer (UI), application layer (Flask backend), and data layer (SQLite with SQLAlchemy ORM)). Basic modules such as user registration, user login, and the core interface were created and tested independently to form a solid base. In later versions, future updates involved installing more important features, such as email and URL phishing security, based on deep learning and machine learning systems. AJAX-based communication was used to seamlessly integrate these modules into the backend, providing real-time feedback without page reloads. The two models, which are the hybrid model trained independently and the validated model, were implemented under the application logic since they provide realistic and quick predictions.

The last increment was dedicated to improving the system's usability and the administration's control. User feedback submission, management of the administrator's inbox, and future support for the models were introduced. Work on each step was then preceded by specific testing and optimization, so that the system could be continually improved without sacrificing scalability, security, or user convenience.

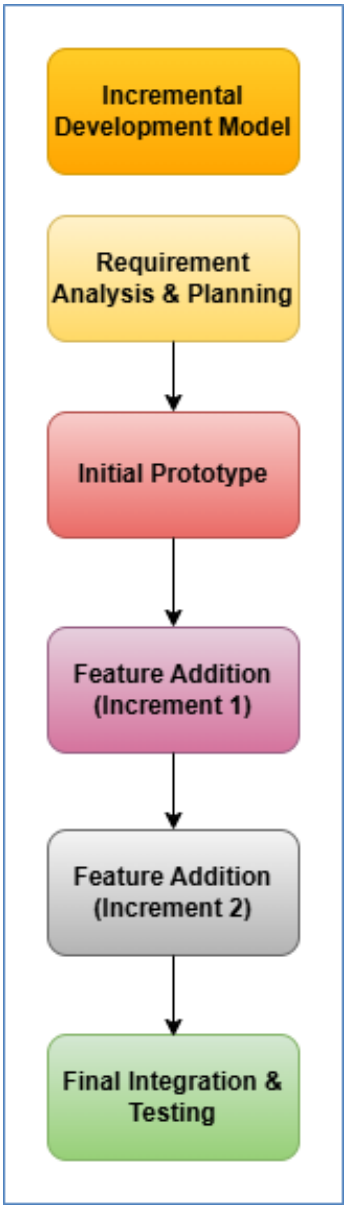

Fig. 10 *Incremental development model*

I. Requirement identification

The purpose of this system is to detect phishing attempts in emails and URLs through a hybrid deep learning and machine learning model. It allows users to submit suspicious content via a web interface and receive real-time predictions, while also enabling admin monitoring and model management.

J. Functional requirements

TABLE IV
FUNCTIONAL REQUIREMENTS

ID	Functional Requirement
FR1	The system must allow users to register and log in securely.
FR2	Users must be able to submit an email (header + body) for phishing detection.
FR3	Users must be able to submit a URL for phishing detection.
FR4	The system must display the detection result using the hybrid model.
FR5	Users must be able to submit feedback on prediction results.
FR6	The admin must be able to log in securely.
FR7	The admin must have access to a dashboard displaying user logs, prediction history, and feedback.
FR8	The admin must be able to view and manage user messages submitted through the contact form.
FR9	The admin must have the ability to upload/retrain the detection models.
FR10	The system must store all relevant data including user info, scan results, and feedback using SQLite via SQLAlchemy.

K. Non-functional requirements

TABLE V
NON-FUNCTIONAL REQUIREMENTS

ID	Non-Functional Requirement
NFR1	The system must return prediction results in under 2 seconds for each scan.
NFR2	The frontend must be responsive across devices (desktop, tablet, mobile).
NFR3	All user data must be stored securely with proper form validation and session management.
NFR4	The application should be easy to maintain and extend, following three-layer architecture.
NFR5	The system should support up to 50 concurrent users without performance degradation.
NFR6	The backend must be built with Python and Flask, using modular code structure.
NFR7	Email and URL scan results must be logged for audit and improvement.

V. CONCLUSION

This study designed and built an intelligent system, PhishGuard, that leverages the effectiveness of Natural Language Processing (NLP) and Deep Learning (DL) to detect phishing attacks in emails and links. The system addresses the shortcomings of conventional heuristic-based models by leveraging complex text analysis, feature extraction, and hybrid classification, resulting in higher accuracy and adaptability while reducing false-positive rates. Developed using the Incremental Development Model, PhishGuard has a modular structure, with a user interface built on Flask that is responsive and secure, and real-time prediction APIs. The fact that extensive unit, system, and user testing have been conducted demonstrates that the system works reliably in the real world. The model yields strong results across various evaluation metrics, such as precision, recall, and confidence scores. Beyond its technical significance, the system supports SDG 16: Peace, Justice, and Strong Institutions by reducing the likelihood of cybercrime, strengthening trust in online communication channels, and fostering safer engagement in online activities. PhishGuard enables users and organizations to access optimal cybersecurity tools that create safer, less opaque digital spaces, crucial to ethical governance and to instilling resilience in institutions.

References

[1] P. Chandre, P. Sontakke, R. Patil, B. D. Shendkar, V. Vanarote, and D. Dhotre, "Mitigating scams, phishing, and malicious attacks: Strategies for enhancing cybersecurity and personal protection," in *International Conference on Smart Trends for Information Technology and Computer Communications*. Springer, 2025. doi: 10.1007/978-981-96-7505-0_35 pp. 447- 463.

[2] E. Kritika, "A comprehensive literature review on phishing url detection using deep learning techniques," *Journal of Cyber Security Technology*, vol. 9, no. 4, pp. 315-343, 2025. doi: 10.1080/23742917.2024.2378552

[3] T. Peng, I. Harris, and Y. Sawa, "Detecting phishing attacks using natural language processing and machine learning," in *2018 ieee 12th international conference on semantic computing (icsc)*. IEEE, 2018. doi: 10.1109/icsc.2018.00056 pp. 300-301.

[4] A. I. Sani, "Phishing detection using natural language processing and behavioural analysis: A multi-faceted approach," *International Journal of Scientific and Management Research*, 2025. doi: 10.37502/ijsmr.2025.81105

[5] B. Abdullahi, "Impact of n-power program on poverty alleviation in nigeria: A study of gombe state," *Journal of Advances in Humanities and Social Sciences*, vol. 11, no. 2, pp. 65-71, 2025. doi: 10.20474/jahss-11.2.1

[6] B. Riskhan, R. Alimam, B. S. Mahaba, M. Gheisari, S. R. Sindiramutty *et al.*, "Enhancing diagnostic accuracy for breast cancer using classical-quantum hybrid and transfer learning technique," *Journal of Soft Computing and Data Mining*, vol. 6, no. 2, pp. 41-56, 2025. doi: 10.30880/jscdm.2022.03.01.004

[7] 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. 65 703-65 727, 2022. doi: 10.1109/access.2022.3183083

[8] A. Alhogail and A. Alsabih, "Applying machine learning and natural language processing to detect phishing email," *Computers & Security*, vol. 110, p. 102414, 2021. doi: 10.1016/j.cose.2021.102414

[9] L. ZhengJie, M. Marjani, S. R. A. Sindiramutty, D. Javaid *et al.*, "Development of an ai-based speech recognition system for chinese dialects: A case study," in *2025 International Conference on Emerging Trends in Networks and Computer Communications (ETNCC)*. IEEE, 2025. doi: 10.1109/etncc66224.2025.11299727 pp. 625-631.

[10] N. Mohamed, "Artificial intelligence and machine learning in cybersecurity: A deep dive into state-of-the-art techniques and future paradigms," *Knowledge and Information Systems*, vol. 67, no. 8, pp. 6969-7055, 2025. doi: 10.1007/s10115-025-02429-y

[11] R. Verma, N. K. Shashidhar, N. Hossain, and N. Rai, "Automatic phishing email detection based on natural language processing techniques," Sep. 3 2019, uS Patent 10,404,745.

[12] F. Tiezhi, S. R. Sindiramutty, A. Khan, A. P. Junfithrana *et al.*, "Web page text recognition and extraction based on ocr," in *2025 International Conference on Metaverse and Current Trends in Computing (ICMCTC)*. IEEE, 2025. doi: 10.1109/icmctc62214.2025.11196233 pp. 1-8.

[13] A. S. Shah, A. Maqsood, A. Shah, M. A. K. Khani, J. Anjum, and S. Zafar, ""enhanced airport operations: Automated baggage drop-off and boarding pass generation for travelers,"," *Journal of Advances in Technology and Engineering Research*, vol. 10, no. 2, pp. 1-6, 2024. doi: 10.20474/jater-10.2.1

[14] F. N. Sanjaya, S. Balakrishnan, S. R. Sindiramutty, and P. Narputro, "Xai technology in eeg signal based disease detection models," in *2025 International Conference on Metaverse and Current Trends in Computing (ICMCTC)*. IEEE, 2025. doi: 10.1109/icmctc62214.2025.11196271 pp. 1-7.

[15] O. Y. Al-Jarrah, C. Maple, M. Dianati, D. Oxtoby, and A. Mouzakitis, "Intrusion detection systems for intra-vehicle networks: A review," *IEEE Access*, vol. 7, pp. 21 266-21 289, 2019. doi: 10.1109/access.2019.2892066

[16] F. Çolhak, M. I. Ecevit, B. E. Uçar, R. Creutzburg, and H. Dağ, "Phishing website detection through multi-model analysis of html content," in *International Conference on Theoretical and Applied Computing*. Springer, 2024. doi: 10.1007/978-981-97-6957-5_15 pp. 171-184.

[17] F. A. Jam, T. I. Khan, and J. Paul, "Driving brand evangelism by unleashing the power of branding and sales management practices," *Journal of Business Research*, vol. 190, p. 115214, 2025. doi: 10.1016/j.jbusres.2025.115214

[18] A. El Aassal, S. Baki, A. Das, and R. M. Verma, "An in-depth benchmarking and evaluation of phishing detection research for security needs," *IEEE Access*, vol. 8, pp. 22 170-22 192, 2020. doi: 10.1109/access.2020.2969780

[19] Z. Kun, S. R. A. Sindiramutty, N. H. B. Bakar, N. Jhanjhi, and F. Ashfaq, "Music generation and key standardization using lstm model," in *2025 International Conference on Emerging Trends in Networks and Computer Communications (ETNCC)*. IEEE, 2025. doi: 10.1109/etncc66224.2025.11299766 pp. 695-700.

[20] M. Korkmaz, O. K. Sahingoz, and B. Diri, "Detection of phishing websites by using machine learning-based url analysis," in *2020 11th international conference on computing, communication and networking technologies (ICCCNT)*. IEEE, 2020. doi: 10.1109/icccnt49239.2020.9225561 pp. 1-7.

[21] X. Weiqi, S. T. C. Hooi, S. R. A. Sindiramutty, D. A. Asirvatham, D. Kumar, and S. Verma, "Surface anomaly detection using machine learning technique," in *2024 International Conference on Emerging Trends in Networks and Computer Communications (ETNCC)*. IEEE, 2024. doi: 10.1109/etncc63262.2024.10767562 pp. 1-7.

[22] G. S. Nayak, B. Muniyal, and M. C. Belavagi, "Enhancing phishing detection: A machine learning approach with feature selection and deep learning models," *IEEE Access*, 2025. doi: 10.1109/access.2025.3543738

[23] S. Jacob, M. Alagirisamy, C. Xi, V. Balasubramanian, R. Srinivasan, N. Jhanjhi, S. M. Islam *et al.*, "Ai and iot-enabled smart exoskeleton system for rehabilitation of paralyzed people in connected communities," *Ieee Access*, vol. 9, pp. 80 340-80 350, 2021. doi: 10.1109/access.2021.3083093

[24] A. Bano, S. H. Hamzah, and E. B. Hafiz, "Interplay between gender & and influencing factors to determine physical activity of school children," *Journal of Management Practices, Humanities and Social Sciences*, vol. 9, no. 2, pp. 120-135, 2025. doi: 10.33152/jmphss-9.2.5

[25] A. Chaudhuri, S. Sarkar, and P. K. Bala, "Thematic exploration and analysis of cybersecurity policies of businesses: An nlp-based approach," *Journal of Organizational Computing and Electronic Commerce*, vol. 35, no. 2, pp. 157-187, 2025. doi: 10.1080/10919392.2024.2435115

[26] T. Koide, N. Fukushi, H. Nakano, and D. Chiba, "Chatspamdetector: Leveraging large language models for effective phishing email detection," in *International Conference on Security and Privacy in Communication Systems*. Springer, 2024, pp. 297-319.

[27] N. Jhanjhi, "Investigating the influence of loss functions on the performance and interpretability of machine learning models," in *International Conference on Mathematical Modeling and Computational Science*. Springer, 2025. doi: 10.1007/978-3-031-91005-0_43 pp. 480-491.

[28] N. Saba, K. Anjali, A. Tanu, A. Porwal, A. Tejaswi, and R. S. Rajendran, "Nlp in social media data processing," in *AI Techniques for Multimedia Data Processing*. IGI Global Scientific Publishing, 2025, pp. 181-226.

[29] M. I. Khalil, M. Humayun, N. Jhanjhi, M. Talib, and T. A. Tabbakh, "Multi-class segmentation of organ at risk from abdominal ct images: A deep learning approach," in *Intelligent Computing and Innovation on Data Science: Proceedings of ICTIDS 2021*. Springer, 2021, pp. 425-434.

[30] Gagandeep and J. Verma, "Natural language processing for sentiment analysis in social media posts to identify suspicious behaviour," *Abhigyan*, vol. 43, no. 2, pp. 143-160, 2025. doi: 10.1177/09702385241284879

[31] A. Kumar, J. M. Chatterjee, V. G. Díaz *et al.*, "A novel hybrid approach of svm combined with nlp and probabilistic neural network for email phishing," *International Journal of Electrical and Computer Engineering*, vol. 10, no. 1, p. 486, 2020. doi: 10.11591/ijece.v10i1.pp486-493

[32] E. Benavides-Astudillo, W. Fuertes, S. Sanchez-Gordon, D. Nuñez-Agurto, and G. Rodríguez-Galán, "A phishing-attack-detection model using natural language processing and deep learning," *Applied Sciences*, vol. 13, no. 9, p. 5275, 2023. doi: 10.3390/app13095275

[33] G. Loveleen, B. Mohan, B. S. Shikhar, J. Nz, M. Shorfuzzaman, and M. Masud, "Explanation-driven hci model to examine the mini-mental state for alzheimer's disease," *ACM Transactions on Multimedia Computing, Communications and Applications*, vol. 20, no. 2, pp. 1-16, 2023. doi: 10.1145/3527174

[34] A. Ozcan, C. Catal, E. Donmez, and B. Senturk, "A hybrid dnn–lstm model for detecting phishing urls," *Neural Computing and Applications*, vol. 35, no. 7, pp. 4957-4973, 2023. doi: 10.1007/s00521-021-06401-z

[35] L. Zhang and P. Zhang, "Phishtrim: Fast and adaptive phishing detection based on deep representation learning," in *2020 IEEE International Conference on Web Services (ICWS)*. IEEE, 2020. doi: 10.1109/icws49710.2020.00030 pp. 176-180.

[36] A. Aljofey, Q. Jiang, Q. Qu, M. Huang, and J.-P. Niyigena, "An effective phishing detection model based on character level convolutional neural network from url," *Electronics*, vol. 9, no. 9, p. 1514, 2020. doi: 10.3390/electronics9091514

[37] M. Elsadig, A. O. Ibrahim, S. Basheer, M. A. Alohali, S. Alshunaifi, H. Alqahtani, N. Alharbi, and W. Nagmeldin, "Intelligent deep machine learning cyber phishing url detection based on bert features extraction," *Electronics*, vol. 11, no. 22, p. 3647, 2022. doi: 10.3390/electronics11223647

[38] P. Maneriker, J. W. Stokes, E. G. Lazo, D. Carutasu, F. Tajaddodianfar, and A. Gururajan, "Urltran: Improving phishing url detection using transformers," in *Milcom 2021-2021 ieee military communications conference (milcom)*. IEEE, 2021. doi: 10.1109/milcom52596.2021.9653028 pp. 197-204.

[39] H. Shirazi, S. R. Muramudalige, I. Ray, and A. P. Jayasumana, "Improved phishing detection algorithms using adversarial autoencoder synthesized data," in *2020 ieee 45th conference on local computer networks (lcn)*. IEEE, 2020. doi: 10.1109/lcn48667.2020.9314775 pp. 24-32.

[40] K. Haynes, H. Shirazi, and I. Ray, "Lightweight url-based phishing detection using natural language processing transformers for mobile devices," *Procedia Computer Science*, vol. 191, pp. 127-134, 2021. doi: 10.1016/j.procs.2021.07.040

[41] P. Verma, A. Goyal, and Y. Gigras, "Email phishing: Text classification using natural language processing," *Computer Science and Information Technologies*, vol. 1, no. 1, pp. 1-12, 2020. doi: 10.11591/csit.v1i1.p1-12

[42] S. H. Gill, M. A. Razzaq, M. Ahmad, F. M. Almansour, I. U. Haq, N. Jhanjhi, M. Z. Alam, and M. Masud, "Security and privacy aspects of cloud computing: A smart campus case study," *Intelligent Automation & Soft Computing*, vol. 31, no. 1, pp. 117-128, 2022. doi: 10.32604/iasc.2022.016597