PhishHook: Catching Phishing Schemes Using Machine Learning

Aliza Raza, Muhammad Sarwar, Jameel Ahmad, Muhammad Husnain Ashfaq

University of Management and Technology, Lahore, Pakistan

Correspondence:

Aliza Raza: F2021266294@umt.edu.pk

Article Link: https://journals.brainetwork.org/index.php/jcai/article/view/83

DOI: https://doi.org/10.69591/jcai.2.2.1

Volume 2, Issue 2, 2024

Funding
No

Citation: A. Raza, M. Sarwar, J. Ahmad, and M. H. Ashfaq, "PhishHook: Catching Phishing Schemes Using Machine Learning," *Journal of Computing and Artificial Intelligence*, vol. 2, no. 2, pp. 1–27, 2024.

Conflict of Interest: Authors declared no Conflict of Interest

Acknowledgment: No administrative and technical support was taken for this research

Article History

Submitted: Sep 11, 2024
Last Revised: Oct 20, 2024
Accepted: Dec 15, 2024

An official Publication of
Beyond Research Advancement & Innovation Network, Islamabad, Pakistan

PhishHook: Catching Phishing Schemes Using Machine Learning

Aliza Raza[*1], Muhammad Sarwar[1], Jameel Ahmad[1], Muhammad Husnain Ashfaq[1]
[1]University of Management and Technology, Lahore, Pakistan

Abstract

Phishing attacks remain a formidable threat in today's digital landscape, posing significant risks to both individuals and organizations. The ever-evolving nature of these attacks outpaces conventional detection methods, necessitating innovative solutions. This paper introduces a cutting-edge dual-layer model for phishing detection, leveraging the combined power of Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks on raw URL data. Our approach begins with the meticulous collection and cleansing of diverse, up-to-date URL datasets to ensure a comprehensive foundation for analysis. The custom-designed CNN extracts spatial patterns inherent in phishing URLs, while the LSTM network captures temporal dependencies and contextual nuances, significantly enhancing detection accuracy. This hybrid model achieves an impressive 98% accuracy, outperforming traditional machine learning techniques in both precision and recall. Extensive experimentation confirms the superiority of our model, which not only minimizes false positives and negatives but also maintains computational efficiency—making it suitable for real-time deployment. The study underscores the critical need for continuous dataset updates and model retraining to keep pace with emerging threats, ensuring robust protection in an increasingly perilous cyberspace. This work represents a significant advancement in phishing detection, offering a scalable, high-performance solution that addresses the challenges of today's dynamic threat environment.

Keywords*:* ML (Machine Learning), SVM (Support Vector Machine), CNN (Convolutional Neural Network), NB (Naïve Bayes), RF (Random Forest), URL (Universal Resource Locator)

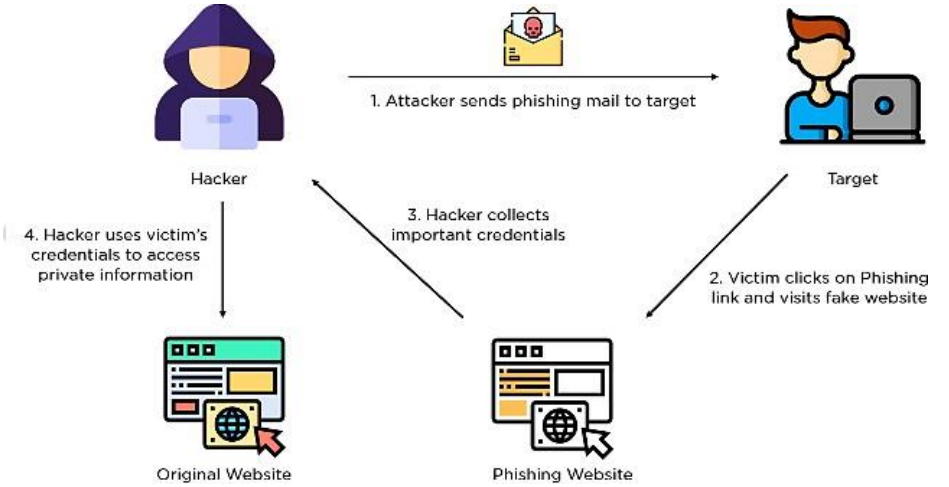
Introduction

In an era dominated by the interconnected web, the internet serves as both a vast repository of information and a complex network linking individuals across the globe. However, this connectivity has also given rise to a nefarious digital underworld, where cybercriminals exploit users' trust through deceptive tactics. One of the most insidious forms of cybercrime is phishing a technique in which attackers create fraudulent websites or emails to trick individuals into revealing sensitive information, such as

[*] Corresponding Author: F2021266294@umt.edu.pk

passwords, financial details, or personal data. Figure 1 outlines the stages of a phishing attack, showing how such an attack is carried out.

Figure 1: *Phishing Attack Demonstration*



As our reliance on online platforms intensifies, so does the sophistication and volume of phishing attacks [1]. The situation worsened during the COVID-19 pandemic, with Gmail intercepting 18 million phishing emails out of 100 million spam messages daily in 2020 [2]. Businesses and governments have suffered significant financial losses due to the growing number of attacks each year. The average cost of a data breach caused by phishing for an organization is approximately $4 million [3], even with existing blacklist and whitelist protection methods in place [4].

The global cost of cybercrime reached $8.44 trillion in 2022 and is projected to rise to $23.84 trillion by 2027 [5]. In 2021 alone, over 1 billion email accounts were exposed, affecting one in five internet users [6]. These alarming statistics underscore the urgent need for an accurate, simple, and reliable tool to detect phishing attacks.

Phishing detection is a particularly compelling area within computer security, where machine learning an AI-related field can significantly enhance existing solutions. Unlike traditional rule-based methods, machine learning models are generally more accurate and efficient, as they can process large volumes of data and perform pattern matching to identify suspicious indicators. Figure 2 shows the yearly increase in the cost of cybercrime.

Figure 2: *Phishing attacks cost per year*

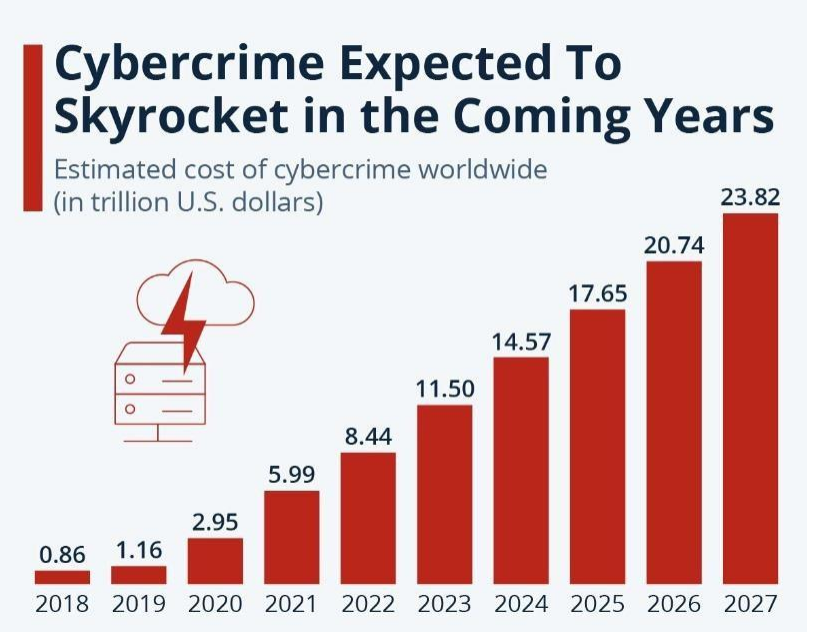
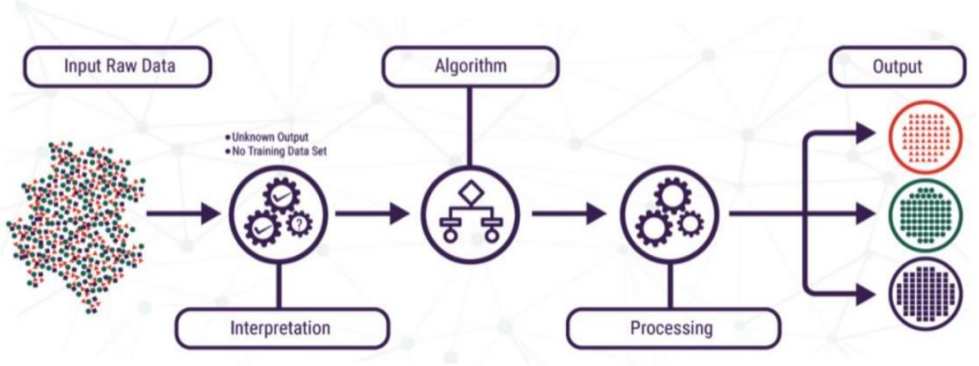


Figure 3: *Unsupervised learning*



These models can learn from past attacks and adapt their strategies, accordingly, making them well-suited to counter evolving cyber threats [7]. In particular, unsupervised learning can be especially effective for phishing detection, as it does not require manual feature extraction or the assignment of weights to inputs. Most of the heavy lifting is done by the model itself: it receives unlabeled data, identifies patterns,

and makes informed decisions based on what it learns. Figure 3 illustrates the transformation of raw data through interpretation and processing, leading to the final outcome.

Recent research in phishing detection has led to the development of various technical approaches, such as logistic regression [8], CNN-based models for detecting phishing emails and websites [9], and feature-based classification methods [10]. For instance, [10] extracted eight features from webpage content to feed into a classification-based algorithm (CBA). The proposed model in [10] was a multi-layered architecture that leveraged both domain-level and DNS packet-level information to generate static and dynamic features.

However, many of these approaches rely on hand-crafted features, which require extensive domain expertise and manual effort. Therefore, the purpose of this paper is to provide further insight into how cybersecurity threats particularly phishing can be addressed more effectively using machine learning techniques, especially those that reduce the need for manual feature engineering.

Background

For an in-depth review of this paper, it is crucial to understand the devastating and harmful impact of phishing, which highlights the urgent need for rapid advancements in the cybersecurity landscape [13].

Phishing attacks have evolved significantly since their first reported instance in 1990, yet the core objective remains the same: to deceive unsuspecting internet users into divulging sensitive information by impersonating legitimate websites or institutions. Phishers often employ social engineering techniques, typically redirecting users to malicious websites via embedded links in emails. However, the attack vectors have expanded to include other channels such as VoIP, SMS, and instant messaging (IM).

A notable evolution is the shift from traditional mass phishing, where emails are sent indiscriminately to more targeted efforts known as spear phishing, in which attackers focus on specific individuals or organizations [13].

While human error continues to be a major factor in the success of phishing scams, this does not diminish the need to develop new methods to safeguard digital infrastructures. In 2020 alone, phishing was reported as the most common type of security breach in the UK and the third most common in Pakistan [14].

Technical Concepts at Hand:

URLs (Uniform Resource Locators): URLs are the addresses used to access resources on the internet. Phishing URLs are specifically crafted to closely resemble legitimate ones, often containing subtle differences intended to deceive users. For example, a phishing URL might use "Daraz.com" instead of the legitimate "daraz.com," exploiting capitalization or slight spelling variations to trick unsuspecting users.

Embedding Layers:

In deep learning, embedding layers are used to convert categorical data—such as words or tokens—into dense vector representations. These embeddings capture semantic relationships and contextual similarities between categories, allowing the model to interpret and learn from the data more effectively.

Convolutional Neural Networks (CNNs):

Convolutional Neural Networks (CNNs) are a type of neural network particularly effective at recognizing patterns in spatial data. When applied to URLs, CNNs can identify common substrings or character sequences that are indicative of phishing attempts, enabling the model to detect malicious patterns embedded within the URL structure.

Long Short-Term Memory (LSTM) Networks:

Long Short-Term Memory networks (LSTMs) are a type of recurrent neural network (RNN) designed to retain information over long sequences. They are particularly effective in capturing the context and sequential structure of data, which is crucial for analyzing URLs and distinguishing between phishing and legitimate websites.

The fundamental theoretical principle behind using machine learning and deep learning for phishing detection lies in identifying patterns and anomalies. Phishing URLs often exhibit one or more distinguishing characteristics that set them apart from genuine URLs, such as:

1. **Pattern Recognition:**

Given a large volume of URLs, machine learning models can identify specific features commonly associated with phishing attempts. These patterns may include unusual characteristics in domain names, irregularities in subdomains, or lexical structures that are atypical of legitimate URLs.

2. **Anomaly Detection:**

Phishing URLs are crafted to mimic legitimate ones but often exhibit structural differences. Detecting URLs that deviate from typical patterns is a key strategy in phishing detection. Machine learning models especially deep learning approaches are

well-suited for this task, as they can automatically identify such anomalies without the need for extensive manual pre-processing or feature engineering.

3. **Data Availability:**

There are two key preconditions that make machine learning models heavily dependent on training data: the availability of data and its quality. For effective phishing URL detection, the dataset must include a large and diverse collection of URLs both those associated with phishing attacks and those linked to legitimate sites. A well-balanced and realistic dataset enables the model to learn meaningful distinctions between malicious and benign URLs. Furthermore, including a wide variety of phishing techniques in the dataset helps ensure that the model can adapt to evolving threats and improve its detection performance over time.

Computational Resources:

Deep learning models such as CNNs and LSTMs require substantial computational resources and time to train. This includes high-performance CPUs or GPUs, as well as sufficient memory and storage capacity.

Real-Time Detection: For practical use, the model must be capable of identifying phishing URLs in real-time or near real-time, enabling effective deployment in real-world scenarios. To achieve this, the model must be optimized to minimize the number of inference steps while maintaining a high level of accuracy.

Generalization:

In other words, the model must be evaluated using previously unseen URLs that were not part of the training data. This ensures that the model not only performs well on the training set but can also accurately predict new malicious URLs that phishing sites may use in the real world.

Adversarial Attacks:

Attackers continuously vary the form and content of their attacks to evade detection. Therefore, the model must be robust against adversarial URLs created by attackers to bypass the detector.

Privacy and Security: When dealing with real-world data, especially beyond academic settings, privacy and security become critical concerns. The system should protect sensitive data and be designed to prevent any unauthorized access or potential intrusions that could compromise its integrity.

Literature Review

The detection of phishing attacks, particularly through URLs and emails, remains a critical area of research in cybersecurity. Various methodologies have been developed to improve the accuracy and efficiency of phishing detection systems. This

literature review examines several significant contributions to this field, focusing on advancements made since 2020.

Phishing URL detection has been approached using numerous methodologies with notable results in recent years. Verma and Das [15] employed a fast feature extraction method involving both lexical and host-based features, achieving an accuracy of 93.8% on a dataset of 2.4 million URLs. In contrast, Nagy et al. [16] evaluated several machine learning models including Random Forest (RF), Naive Bayes (NB), CNN, and LSTM—and reported the highest accuracies of 95.4% from RF and 96.01% from NB on a dataset of 1.2 million URLs. Rasymas and Dovydaitis [17] applied a deep learning approach combining CNN and LSTM layers, achieving 94.1% accuracy on a dataset of 2.5 million URLs.

Further improvements were made by Prabakaran et al. [18], who introduced a novel phishing detection mechanism using Variational Autoencoders (VAE) to extract features directly from raw URLs. This approach achieved a higher accuracy of 97% using a smaller dataset of 100,000 URLs. Similarly, Zhou et al. [19] applied the LightGBM model with domain name features, achieving 93% accuracy on 24,000 URLs, with an overall accuracy of 88%. Dutta [20] utilized Recurrent Neural Networks (RNN) for phishing detection, reporting 97% accuracy and an F1 score of 96.4% on a relatively small dataset of 13,700 URLs.

Other notable works include Adebowale et al. [21], who developed hybrid classification models combining CNN and LSTM for image-based phishing detection, achieving 93.28% accuracy on a dataset comprising 1 million URLs and 10,000 images. Jamal et al. [22] proposed an enhanced Transformer model leveraging self-attention mechanisms, fine-tuning a DistilBERT model to achieve 97% validation accuracy and 97.1% test accuracy on a balanced dataset of 4,825 URLs. Lastly, Alnemmari and Alshimmari [24] evaluated several models with varying performances: ANN (93%), SVM (93.1%), Decision Tree (94%), and Random Forest (97%) on a dataset of 11,055 samples. ALSUBAEI et al. [23] proposed a hybrid deep learning framework, though specific details on accuracy and datasets were not provided. Table 1 presents the different approaches, their corresponding datasets, and their accuracies.

Table 1: *Unsupervised learning*

	Research Information		
	Methodology	**Accuracy**	**Dataset Size**
Verma and Das (2017)	Fast feature extraction from URLs using machine learning	93.8%	2.4 million URLs
N. Nagy et al. (2023)	Four Models RF, NB, CNN, and LSTM	93.19% for CNN and 93.21% for LSTM	1.2M

	Research Information		
	Methodology	**Accuracy**	**Dataset Size**
T. Rasymas and L.Dovydaitis (2020)	Deep neural network consisting of multiple CNN and LSTM layers.	94.1%	2.5M
M.K.Prabakaran,P.M. Sundaram, and A.D. Chandrasekar (2023)	In the proposed framework, the inherent feature of a raw URL is directly extracted by the VAE model by reconstructing the original input URL to improve phishing URL identification.	97%	100,000URLs
J. Zhou, H. Cui, X. Li, W. Yang, and X. Wu (2023)	Uses LightGBM classification model after performing feature extraction and feature selection	93.88%	24,000
Dutta (2021)	RNN-based URL detection technique	97.4%	13,700 URLs
Adebowale, M.A., Lwin, K.T. and Hossain, M.A. (2023)	Hybrid classification model using CNN+LSTM for image detection.	93.28%	1M URLs and 10,000 images
S. Jamal, H. Wimmer, and I. H. Sarker (2023)	Utilizes transformer-based self-attention mechanisms to improve pre-trained BERT models. Employs optimization and fine-tuning techniques on DistilBERT and RoBERTA models with imbalanced and balanced datasets.	Achieved validation accuracy of 97.50% and test accuracy of 97.10% for DistilBERT on a balanced dataset	747 spams, 189 phishing, 4825 ham samples.
ALSUBAEI, Almazroi, Ayub(2024)	Hybrid deep learning framework integrating multiple machine learning techniques for phishing detection. Utilizes data preprocessing, SMOTE for balancing datasets, and PCA for feature selection.	Shows effective clustering and separation of phishing and legitimate websites. Evaluated using True Positive and True Negative values with detailed correlation matrices.	Not Specified

Methodology

Suggested Approach

Based on the literature review conducted in this work, several methodologies such as CNNs, LSTMs, and ensemble learning models have proven effective for phishing

detection. Prior studies have demonstrated that these approaches can automatically extract meaningful features from raw URLs and achieve high recall rates in identifying phishing attempts. However, there remains a gap in research involving hybrid models that combine CNNs and LSTMs, especially when applied to large, diverse datasets. Such hybrid approaches have the potential to improve generalization and adapt more effectively to emerging phishing strategies employed by attackers.

This paper addresses this gap by leveraging these insights and proposing a more comprehensive and robust method. Initial experiments were also performed using basic machine learning algorithms including Naïve Bayes, SVM, AdaBoost, Decision Tree, and Random Forest which provided a useful baseline. Nonetheless, manual feature engineering was labor-intensive, and in many cases, the overall performance heavily depended on the quality of these handcrafted features.

Description of Research Design and Procedures Used

Our research design follows a multi-phase approach to develop and evaluate a hybrid deep learning model for phishing detection. The key phases include:

1. Data Collection and Prepossessing:

Data Sources: We collected a diverse dataset from PhishTank, OpenPhish, AlexaRank, and Kaggle, ensuring a balanced mix of phishing and legitimate URLs. This approach addresses the need for a comprehensive dataset, as emphasized in literature.

Preprocessing: URLs are normalized and tokenized before being converted into numerical representations. Padding is applied to ensure a uniform input size for the model. This preprocessing step guarantees that the data is clean and well-structured, facilitating effective learning.

2. Model Development:

We designed a hybrid model combining CNNs and LSTMs, leveraging their complementary strengths. The CNN layers focus on detecting local patterns within URLs, such as characteristic substrings, while the LSTM layers capture sequential dependencies and contextual information across the URL sequence. This hybrid approach builds upon findings from previous studies and aims to improve both feature extraction and context understanding for more accurate phishing detection.

- **Embedding Layer:** This layer transforms tokenized URLs into dense vector representations, effectively capturing semantic information.
- **Convolutional Layers:** Detects local patterns indicative of phishing attempts.
- **LSTM Layers:** Capture long-term dependencies and contextual information.
- **Fully Connected Layers:** Aggregate features and make the final classification.

- **Dropout Layer:** A Dropout layer is added immediately after the Bidirectional LSTM layers to prevent the model from becoming overly dependent on specific neurons. This regularization technique helps improve generalization, making the model better suited for testing on unseen data.

3. **Training and Evaluation:**

- **Training:** Hyperparameters are tuned for optimal performance.
- **Evaluation:** Performance is evaluated using accuracy, precision, recall, and F1-score—metrics that together provide a comprehensive assessment of the model's effectiveness.

Development of an Appropriate Research Strategy

- Our research strategy was developed by addressing gaps identified in the literature and the need for a robust phishing detection system. The key components of our approach include:
- Leveraging Hybrid Models: Our literature review showed that CNNs and LSTMs consistently achieve high accuracy. Combining these two models creates a powerful hybrid that benefits from CNNs' ability to identify local patterns and LSTMs' strength in capturing sequential dependencies, making it well-suited for phishing URL detection.
- Ensuring Data Diversity: We utilize a large and diverse dataset to enhance the model's generalizability. Additionally, maintaining a balanced dataset helps improve detection accuracy across phishing and legitimate URLs.
- Employing Advanced Techniques: A primary goal of this research is to reduce the manual effort involved in feature extraction. Therefore, we focus on modern methods that automatically extract features, eliminating the need for hand-crafted inputs.
- Evaluating Robustly: We apply comprehensive performance metrics to rigorously assess the model's reliability and effectiveness, ensuring its suitability for real-world deployment.

Sources of Data and Sampling Procedures

1. **Sources of Data:**

PhishTank and https://map.httpcs.com/: Provide real-time, community-verified phishing URLs.

AlexaRank [25]: Supplies legitimate URLs to help maintain a balanced dataset.

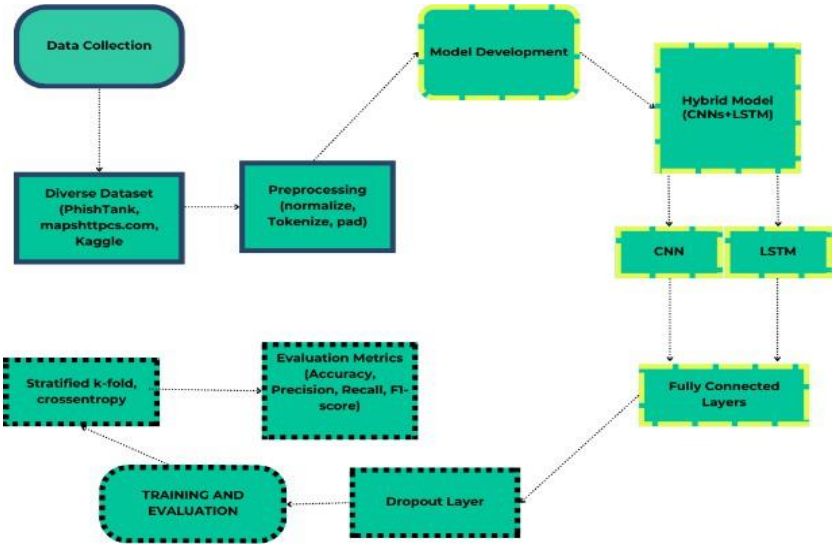
Kaggle [11]: Offers a well-curated dataset containing both legitimate and phishing websites.

2. **Sampling Procedures:**

Stratified Sampling: Ensures proportional representation of both phishing and legitimate URLs in the training and testing datasets, maintaining a balanced distribution.

Data Augmentation: Techniques such as random sampling and oversampling are applied to address class imbalances and improve the model's ability to detect phishing URLs.

Figure 4: *Suggested Approach*



In conclusion, our approach builds on the strengths and addresses the gaps identified in previous research by leveraging a hybrid deep learning model, utilizing a comprehensive dataset, and employing rigorous training and evaluation procedures. This strategy aims to develop a robust, accurate, and generalizable phishing detection system capable of identifying both known and emerging phishing threats. Figure 4 illustrates the approach used to process collected data through a series of analytical techniques.

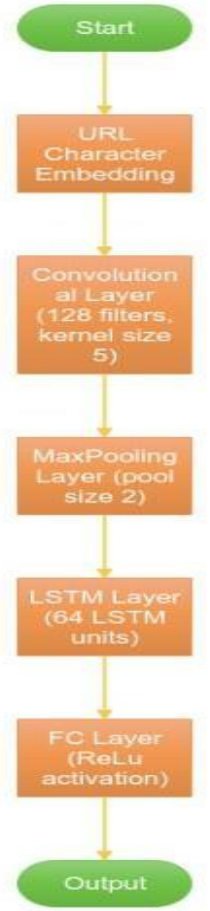
Workflow of the system

Top of Form, Bottom of Form

The primary focus of this study is based on the proven success of hybrid deep learning models in previous research. Combining CNNs with LSTM networks allows

for automated feature extraction while reducing the memory dependency challenges typically associated with LSTMs, making this approach particularly well-suited for phishing URL detection.

Figure 5: *Work Flow Diagram*



In this architecture, CNNs effectively detect local patterns, whereas LSTMs capture the sequential context within URLs. Figure 5 shows which combination is essential for accurately identifying phishing URLs, which often closely resemble legitimate ones and may contain subtle, complex patterns that differentiate them.

Algorithms/Architecture

1. **Embedding Layer**

Purpose: This layer translates discrete URL tokens—whether words or characters into high-level dense vectors. Each unique token in the URL is mapped to a corresponding vector representation.

Function: The embedding layer facilitates semantic understanding by mapping similar tokens to vectors that are close in the vector space. This transformation is crucial for capturing relationships between different components of a URL.

Example: For instance, if the token "login" frequently appears in phishing URLs, the embedding layer helps the model learn its significance within the phishing context, improving detection accuracy.

2. **Convolutional Neural Networks (CNN) Layers:**

Purpose: CNN layers are designed to identify local features such as specific patterns or sequences within the URL embeddings. These layers can detect distinctive substrings or character sequences often associated with phishing scams.

Function:

Convolutional Filters: These filters slide over the input embeddings, detecting various patterns. Each filter can learn to recognize different attributes, such as suspicious keywords or unusual domain structures.

Activation Functions: Non-linear functions like ReLU or sigmoid are applied to introduce non-linearity, enabling the model to capture complex patterns effectively.

Example: A CNN filter might detect the substring "secure" embedded unusually in the middle of a URL, which can be indicative of a phishing attempt.

3. **MaxPooling Layer**

Purpose: This layer reduces the number of feature maps generated by the CNN layers, preserving important features while improving computational efficiency.

Function:

Pooling Operation: MaxPooling selects the maximum value within a specified region of the feature map, highlighting the most prominent presence of a feature in that area.

Example: MaxPooling makes it easier to detect if a specific feature—such as the presence of certain keywords—appears in multiple regions of the feature map and helps identify the strongest signal.

4. **Long Short-Term Memory (LSTM) Layers:**

Purpose: The LSTM layers capture sequential relationships and long-term dependencies within URLs. This is especially important for understanding the order and hierarchy of different URL components.

Function:

Memory Cells: LSTMs have memory cells that retain important information across long sequences, enabling the model to remember key patterns and contexts from earlier in the URL.

Gates: LSTMs use three gates input, forget, and output to regulate information flow. These gates decide what information to keep, discard, or output at each step in the sequence.

Example: An LSTM layer can recognize that tokens like "login" or "account" appearing in sequence within a URL regardless of other intervening characters often indicate phishing intent.

5. **Fully Connected (Dense) Layers:**

Purpose: These layers use the features obtained by the preceding layers to make the final classification to be made.

Function:

Neurons: A fully connected layer entails that every neuron in the new layer gets input from all neurons in the previous layer which helps the model incorporate all learned features.

Activation Functions: Using non-linear activation, such as Rectified Linear Unit (ReLU), is useful for the model to capture intricate decision frontiers.

Example: The dense layers use as inputs the outputs from the CNN and LSTM layers to predict if there are patterns of a phishing URL.

6. **Dropout Layer:**

Purpose: This layer also introduces the concept of dropout; it randomly assigns a fraction of the input units to zero during the training phase to reduce the overfitting of the network.

Function:

Regularization: During training, dropout teaches the algorithm to ignore some neurons this enables generalization since the algorithm is not overly dependent on certain properties.Example: There is also an inherent dropout process that occurs in each iteration of the training process in which certain neurons are excluded while others are included; this kind of strategy makes the model have deeper learning capabilities because it is forced to learn on different features of the data set at different times.

7. **Output Layer:**

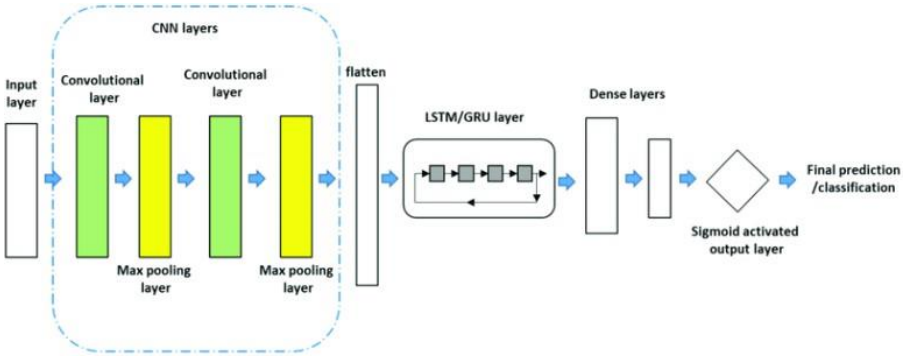
Purpose: This layer provides the final decision of our system in the form of binary classification, that is either its a phishing URL or a legitimate one.

Function:

Sigmoid Activation: The sigmoid activation function gives a normalized output between 0 and 1 with the percentage chance that the URL is phishing. Since the results are continuous values, a certain value (often 0.5) is used to arrive at the final binary decision. The sigmoid function is given by the formula:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

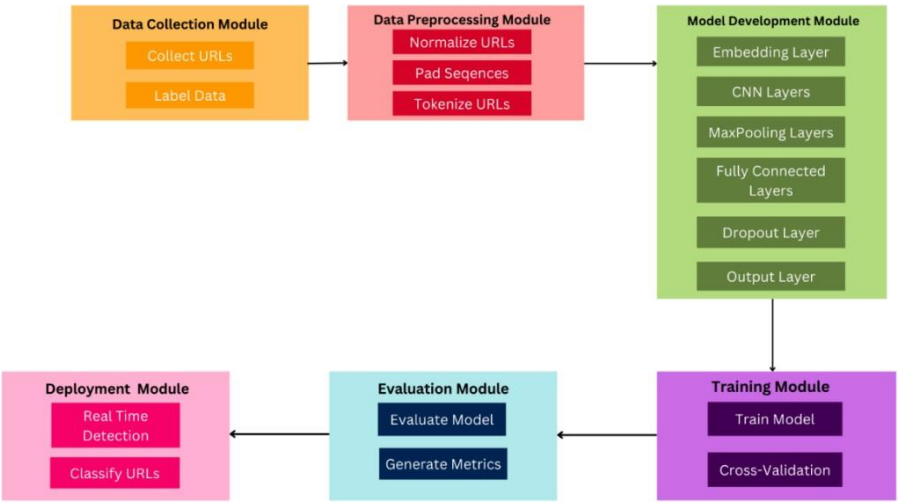
Figure 6: *Unsupervised learning*



Experimentation

Our first batch of experiments used Naïve Bayes, Random Forest, and Decision Tree which yielded great accuracy however, the recall and precision were low on all. Then we moved on to training our actual model and running test experiments on our actual model using our intensive dataset. The dataset was split into training and testing datasets with 80%, and 20% respectively. Figure 6 illustrates the transformation of the input layer into successive layers as the data progresses through the model. Figure 7 shows how different modules are used to process both collected and recent data.

Figure 7: *PhishHook Architecture*



Results And Discussion

Experimental Setup

The most important part of our experimental setup is to fine-tune the hyperparameters. Which are convolutional filters (num_filters=256), output dimensions (embedding_dim=256), with kernel size=8 and lstm_units=128 and dropout_rate=0.3.

We also used the T4 GPU that is available at runtime in Google Colab to have the best resources available which greatly impacted our execution time.

Results

The results of our first batch were quite promising and gave us a baseline to train the model on our actual dataset that contains over a million instances.

1. Evaluation Metrics

The evaluation metrics chosen to give a fleshed-out image of our results were; Accuracy, Precision, recall, and F1-score.

a) **Accuracy:**

Accuracy is used to measure the overall correctness of a model. It describes how many predictions the model can get right out of the whole dataset given. It is described by the no. of true positives, true negatives, false positives, and false negatives. It is given by the formula:

$$\text{Accuracy} = \frac{\text{true positives} + \text{true negatives}}{\text{true positives} + \text{true negatives} + \text{false negatives} + \text{false positives}}$$

Model Accuracy: 98%

b) **Precision:**

It gives the number of true positives out of the whole positives. This is a better metric or a supporting metric to describe your model. It is given by the formula:

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

Model Precision: 99.48 %

c) **Recall:**

Recall is another important metric to measure your model. It shows how good the model is at predicting all positives from the true positives and false negatives included. It is given by the formula:

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

Model Recall: 99.48%

d) **F1-score:**

F1-score provides a balance between precision and recall; therefore it gives us a better understanding of our model. It is a very important evaluation metric. It is given by the formula:

$$F1 = 2 * \frac{\text{precision} * \text{recall}}{\text{Precision} + \text{recall}}$$

Model F1-score: 97%

The execution time was 89.4s, Figure 8 and 9 illustrates the accuracies, with an F1 score of 97%.

Figure 8: *Accuracy precision and Recall*

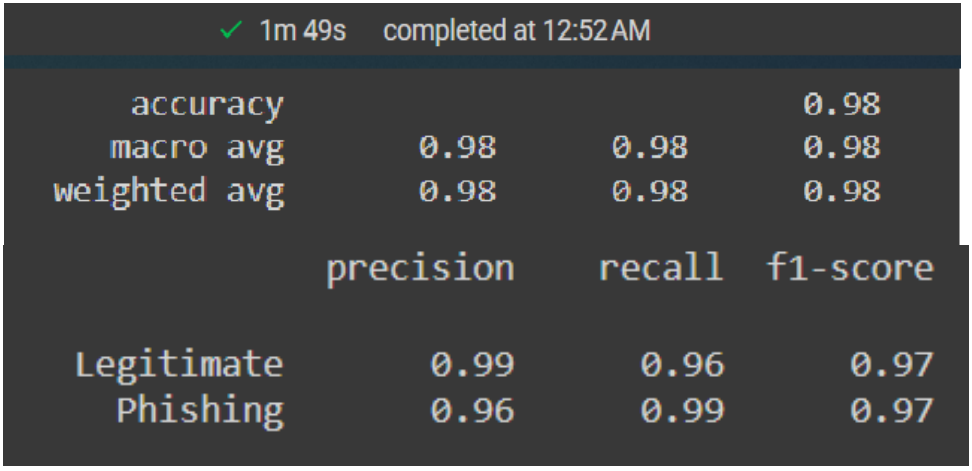
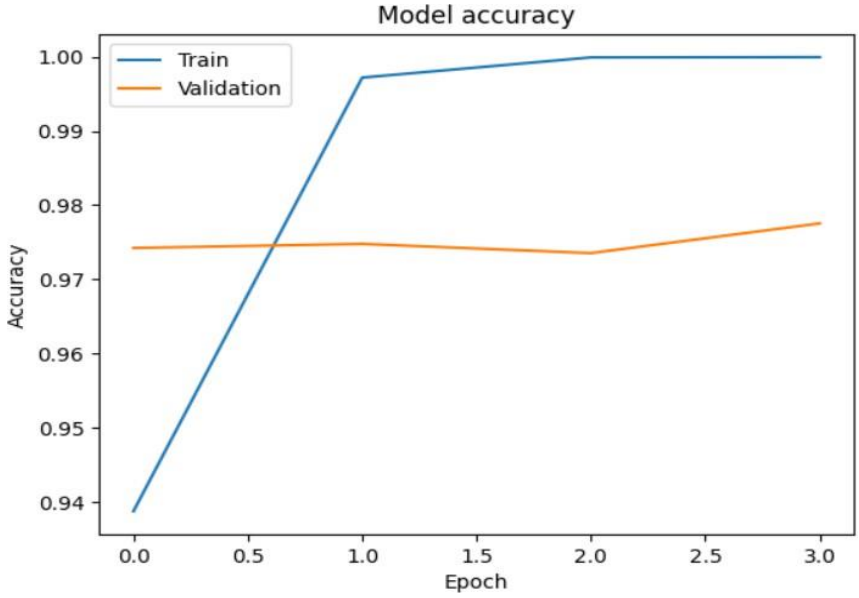
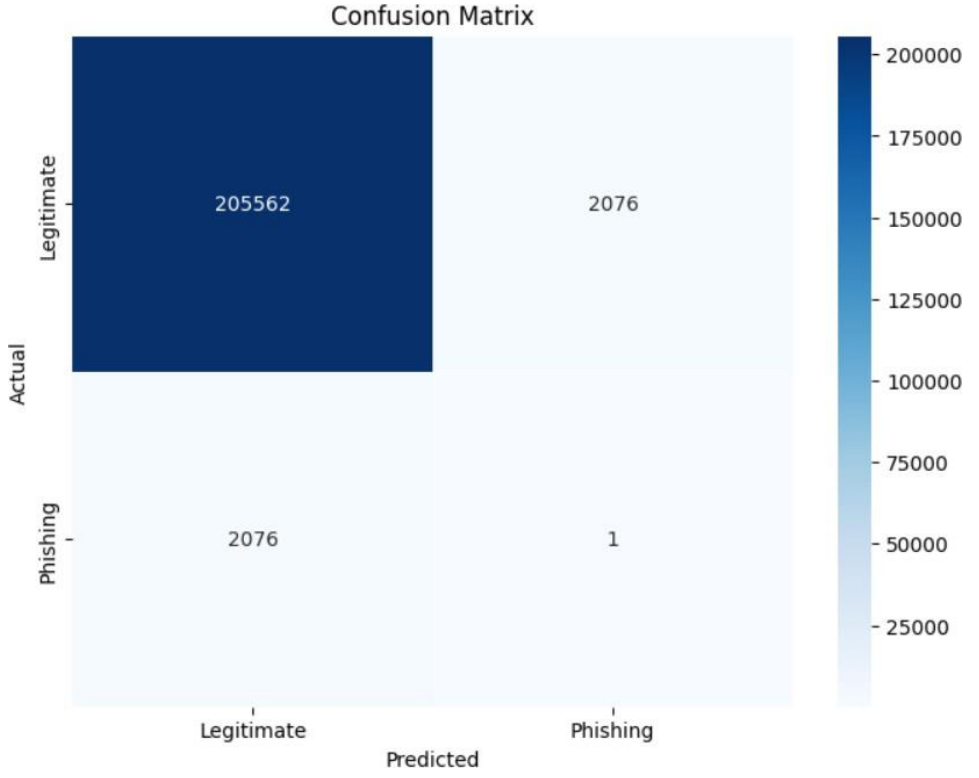


Figure 9: *PhishHook Accuracy Graph*



Out of the 1,048,576 URLs that were tested the number of TP(True Positive) was 1,022,435, TN(True Negative) was 15,452, FP (False Positive) was 5,344, and FN(False Negative) was 5,344. Figure 10 shows the confusion matrix.

Figure 10: *PhishHook Confusion Matrix*



Discussion/Analysis

Below is a table comparing our model's accuracy, precision, recall, and F1-score with several established models, including those from [15], [16], [17], [18], and [19]. We selected these metrics to comprehensively evaluate model performance.

The experimental results demonstrate that our model achieves the highest accuracy among all compared methods. More importantly, our model excels in both precision and recall, resulting in a state-of-the-art phishing detection system.

PhishHook was also deployed locally using Huggingface, an AI model deployment platform, to evaluate its real-time performance. In this online setting, it maintained an average accuracy of 97%, with a response time of just a few seconds.

These performance differences underscore the limitations of traditional machine learning models in addressing the sophisticated and evolving nature of phishing attacks. CNNs, in particular, show superior ability to capture the complex patterns inherent in phishing URLs.

Managing Expectations:

Although the results are very promising and have been successful on all fronts, some limitations remain. Primarily, the dataset for a problem like this must be continuously updated, and the model requires periodic retraining to stay effective. Table 2 provides a comparative analysis of various approaches, including their accuracy, precision, recall, and F1 score.

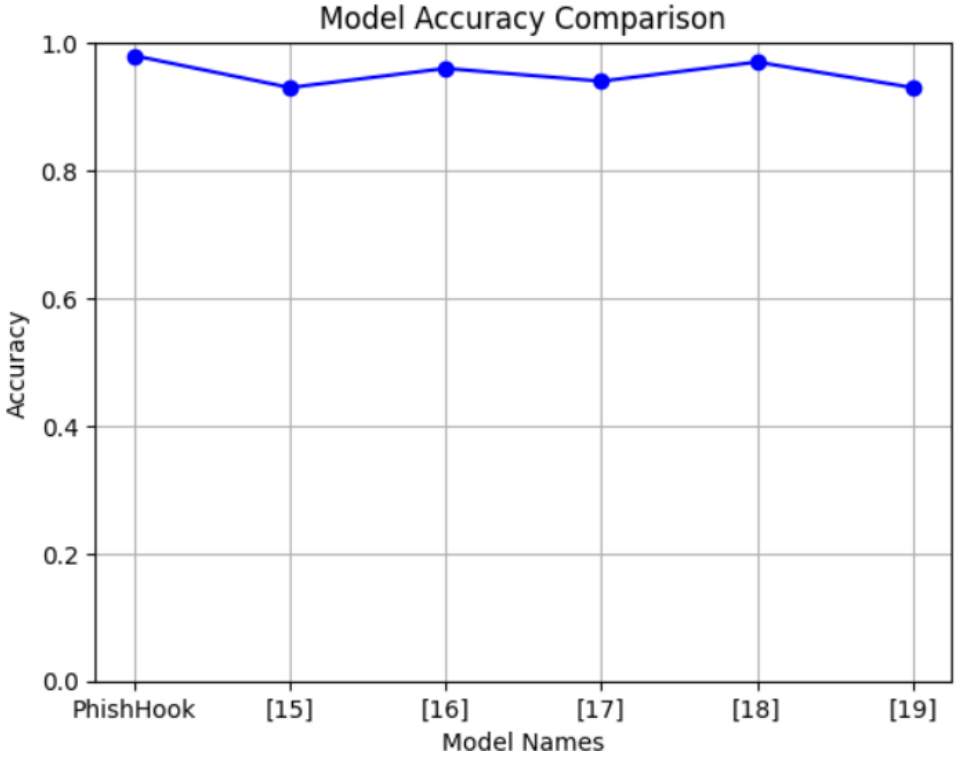
Table 2: *Accuracy Comparison Table with other State of the Art Models*

Model	**Metrics**			
	Accuracy	***Precision***	***Recall***	***F1-score***
PhishHook	98%	99.48%	99.48%	97%
Verma and Das [15]	93.8%	96.44%	93.21%	95.01%
N. Nagy et al.(2023) [16]	Highest of 96.01% NB	Highest of 95.65% NB	Highest of 100% CNN LSTM	Highest of 93.92% NB
T. Rasymas and L. Dovydaitis (2020) [17]	94.1%	97.22%	88.82%	93.79%
M. K. Prabakaran, P. M. Sundaram, and A. D. Chandrasekar (2023) [18]	97%	97.89%	97.20%	97.54%
J. Zhou, H. Cui, X. Li, W. Yang, and X. Wu(2023) [19]	93.88%	94.78%	92.88%	93.82%

Although the results are highly promising and have been successful on all fronts, some limitations remain. Chief among these is the need for continuous dataset updates

and periodic model retraining to maintain effectiveness. Figure 11 illustrates the accuracy comparison among various approaches.

Figure 11: *Line Graph Comparison of Accuracies*



Our model significantly outperforms all other models, as shown in the table above. This improvement can be attributed to several factors, including a better-curated dataset and more effective hyperparameter tuning. It is evident that the model's performance is highly dependent on the quality of the dataset, which contributed to its superior results. Additionally, our model achieved the shortest training time among all, completing it in just 89.4 seconds.

Comparison with Traditional Models:

Our results also demonstrate that the model outperforms traditional machine learning algorithms such as Random Forest, SVM, Decision Tree, Naïve Bayes, and LightGBM. Figure 12 presents a bar graph illustrating the accuracy comparison

between proposed approaches and traditional models. Figure 13 presents a comparative analysis of evaluation metrics against state-of-the-art models.

Figure 12: *Bar Graph of Accuracy Comparison with Traditional Models*

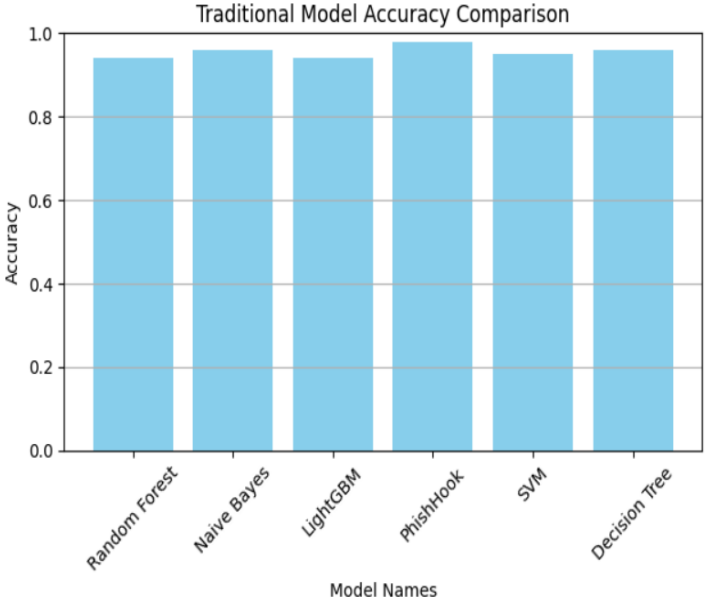
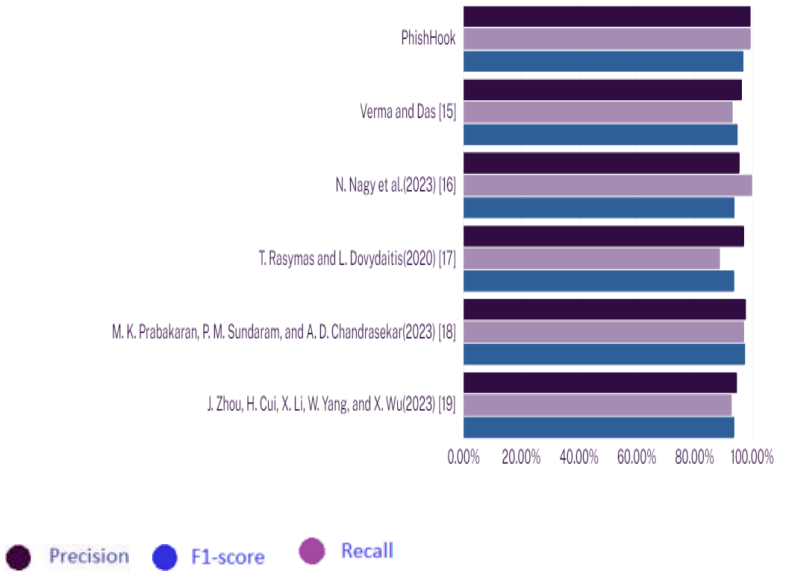


Figure 13: *Evaluation Metrics Comparison with State-of-the-Art Models*



Conclusion

By using the proposed deep learning model, we successfully classify phishing websites with an accuracy of 98%. Our model not only outperforms many existing models in accuracy but also excels in other key performance metrics. Additionally, it requires less training time, indicating reduced computational complexity. These results highlight the importance of a high-quality dataset and fine-tuned hyperparameters in achieving superior accuracy.

Given the increasing prevalence of phishing attacks and cybercrime, it is crucial to continuously improve security measures and detection tools. This paper contributes to that ongoing effort.

Future work will focus on further refining the model architecture to enhance performance, such as incorporating raw HTML data and adapting the approach to detect other phishing variants like spear phishing and whaling. Additionally, future research could involve augmenting the dataset with more diverse samples to cover a broader spectrum of phishing techniques. Exploring automated retraining methods to minimize manual intervention will also be a valuable direction.

References

[1] S. Chaudhuri and U. Dayal, "An overview of data warehousing and OLAP technology," *ACM SIGMOD Rec.*, vol. 26, no. 1, pp. 65–74, Mar. 1997, doi: 10.1145/248603.248616.
[2] W. H. Inmon, *Building the data warehouse*. John wiley & sons, 2005.
[3] S. H. A. El-Sappagh, A. M. A. Hendawi, and A. H. El Bastawissy, "A proposed model for data warehouse ETL processes," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 23, no. 2, pp. 91–104, 2011.
[4] H. Dabbèchi, A. Nabli, and L. Bouzguenda, "Towards Cloud-Based Data Warehouse as a Service for Big Data Analytics," in *Computational Collective Intelligence*, vol. 9876, N. T. Nguyen, L. Iliadis, Y. Manolopoulos, and B. Trawiński, Eds., in Lecture Notes in Computer Science, vol. 9876. , Cham: Springer International Publishing, 2016, pp. 180–189. doi: 10.1007/978-3-319-45246-3_17.
[5] U. Aftab and G. F. Siddiqui, "Big data augmentation with data warehouse: A survey," in *2018 IEEE International Conference on Big Data (Big Data)*, IEEE, 2018, pp. 2785–2794. Available: https://ieeexplore.ieee.org/abstract/document/8622206/
[6] S. Chaudhuri, U. Dayal, and V. Narasayya, "An overview of business intelligence technology," *Commun. ACM*, vol. 54, no. 8, pp. 88–98, Aug. 2011, doi: 10.1145/1978542.1978562.
[7] D. Agrawal, S. Das, and A. El Abbadi, "Big data and cloud computing: current state and future opportunities," in *Proceedings of the 14th International Conference on Extending Database Technology*, Uppsala Sweden: ACM, Mar. 2011, pp. 530–533. doi: 10.1145/1951365.1951432.
[8] L. Duan and L. Da Xu, "Business intelligence for enterprise systems: a survey," *IEEE Trans. Ind. Inform.*, vol. 8, no. 3, pp. 679–687, 2012.
[9] H. Chen, R. H. Chiang, and V. C. Storey, "Business intelligence and analytics: From big data to big impact," *MIS Q.*, pp. 1165–1188, 2012.
[10] A. Cuzzocrea, L. Bellatreche, and I.-Y. Song, "Data warehousing and OLAP over big data: current challenges and future research directions," in *Proceedings of the sixteenth international workshop on Data warehousing and OLAP*, San Francisco California USA: ACM, Oct. 2013, pp. 67–70. doi: 10.1145/2513190.2517828.
[11] E. Abdelaziz and O. Mohamed, "Optimisation of the queries execution plan in cloud data warehouses," in *2015 5th World Congress on Information and Communication Technologies (WICT)*, IEEE, 2015, pp. 219–133. Available: https://ieeexplore.ieee.org/abstract/document/7489659/
[12] D. J. Abadi, "Data management in the cloud: Limitations and opportunities.," *IEEE Data Eng Bull*, vol. 32, no. 1, pp. 3–12, 2009.
[13] M. Brantner, D. Florescu, D. Graf, D. Kossmann, and T. Kraska, "Building a database on S3," in *Proceedings of the 2008 ACM SIGMOD international*

conference on Management of data, Vancouver Canada: ACM, Jun. 2008, pp. 251–264. doi: 10.1145/1376616.1376645.

[14] D. Lomet, A. Fekete, G. Weikum, and M. Zwilling, "Unbundling Transaction Services in the Cloud." arXiv, Sep. 09, 2009.. Available: http://arxiv.org/abs/0909.1768

[15] S. Das, D. Agrawal, and A. El Abbadi, "ElasTraS: An elastic, scalable, and self-managing transactional database for the cloud," *ACM Trans. Database Syst.*, vol. 38, no. 1, pp. 1–45, Apr. 2013, doi: 10.1145/2445583.2445588.

[16] A. Aboulnaga, K. Salem, A. A. Soror, U. F. Minhas, P. Kokosielis, and S. Kamath, "Deploying database appliances in the cloud.," *IEEE Data Eng Bull*, vol. 32, no. 1, pp. 13–20, 2009.

[17] N. Paton, M. A. De Aragão, K. Lee, A. A. Fernandes, and R. Sakellariou, "Optimizing utility in cloud computing through autonomic workload execution," *Bull. Tech. Comm. Data Eng.*, vol. 32, no. 1, pp. 51–58, 2009.

[18] J. Widom, "Research problems in data warehousing," in *Proceedings of the fourth international conference on Information and knowledge management - CIKM '95*, Baltimore, Maryland, United States: ACM Press, 1995, pp. 25–30. doi: 10.1145/221270.221319.

[19] E. Guermazi, M. B. Ayed, and H. Ben-Abdallah, "Adaptive security for Cloud data warehouse as a service," in *2015 IEEE/ACIS 14th International Conference on Computer and Information Science (ICIS)*, IEEE, 2015, pp. 647–650. Available: https://ieeexplore.ieee.org/abstract/document/7166672/

[20] A. R. Nimje, "Data Analytics as a Service (DAaaS): An Arriving Technology in Cloud Computing," *Int. J. Emerg. Trend Eng. Basic Sci.*, vol. 2, no. 1, pp. 181–186, 2015.

[21] S. Kurunji, T. Ge, B. Liu, and C. X. Chen, "Communication cost optimization for cloud Data Warehouse queries," in *4th IEEE International Conference on Cloud Computing Technology and Science Proceedings*, IEEE, 2012, pp. 512–519. Available: https://ieeexplore.ieee.org/abstract/document/6427580/

[22] A. M. Van, H. L. Lv, V. L. Cheng, and F. V. Wang, "Design of cloud data warehouse and its application in smart grid," in *International Conference on Automatic Control and Artificial Intelligence (ACAI 2012)*, Xiamen, China: Institution of Engineering and Technology, 2012, pp. 849–852. doi: 10.1049/cp.2012.1108.

[23] A. Abid *et al.*, "A survey on search results diversification techniques," *Neural Comput. Appl.*, vol. 27, no. 5, pp. 1207–1229, Jul. 2016, doi: 10.1007/s00521-015-1945-5.

[24] M. G. Kahn *et al.*, "Migrating a research data warehouse to a public cloud: challenges and opportunities," *J. Am. Med. Inform. Assoc.*, vol. 29, no. 4, pp. 592–600, 2022.

[25] L. Dinesh and K. G. Devi, "An efficient hybrid optimization of ETL process in data warehouse of cloud architecture," *J. Cloud Comput.*, vol. 13, no. 1, p. 12, Jan. 2024, doi: 10.1186/s13677-023-00571-y.

[26] A. Nambiar and D. Mundra, "An overview of data warehouse and data lake in modern enterprise data management," *Big Data Cogn. Comput.*, vol. 6, no. 4, p. 132, 2022.

[27] S. Rohajawati, "Effectiveness of Digital Transformation in Data Warehouse Technology," *Indones. J. Comput. Sci.*, vol. 13, no. 1, 2024. Available: http://ijcs.stmikindonesia.ac.id/ijcs/index.php/ijcs/article/view/3759

[28] M. Patel and D. B. Patel, "Data Warehouse Modernization Using Document-Oriented ETL Framework for Real Time Analytics," in *Rising Threats in Expert Applications and Solutions*, vol. 434, V. S. Rathore, S. C. Sharma, J. M. R. S. Tavares, C. Moreira, and B. Surendiran, Eds., in Lecture Notes in Networks and Systems, vol. 434. , Singapore: Springer Nature Singapore, 2022, pp. 33–41. doi: 10.1007/978-981-19-1122-4_5.