Vol. 8, No. 1, February 2026

ISI Journal of Information Systems and Informatics

Published By Asosiasi Doktor Sistem Informasi Indonesia

URL-Based Phishing Detection Using a BERT-LSTM Model

Hilman Singgih Wicaksana[1], Usman Ependi[2], Ari Muzakir[3]

[1]Informatics Program, Faculty of Law, Management, and Informatics, Universitas Karya Husada, Semarang, 50276, Indonesia

[2]Informatics Department, Postgraduate Program, Universitas Bina Darma, Palembang, Indonesia

[3]Faculty of Science and Technology, Universitas Bina Darma, Palembang, 30111, Indonesia

Received:
January 26, 2026
Revised:
March 3, 2026
Accepted:
March 18, 2026
Published:
April 3, 2026

Corresponding Author:
Author Name*:
Hilman Singgih Wicaksana
Email*:
singgih.hilman@gmail.com

DOI:
10.63158/journalisi.v8i1.1543

Abstract. The rising prevalence of phishing websites presents substantial cybersecurity threats by deceiving users into revealing sensitive information through malicious URLs. This study aims to enhance phishing URL detection by introducing a deep learning model that combines Bidirectional Encoder Representations from Transformers (BERT) with Long Short-Term Memory (LSTM). In this framework, BERT is fine-tuned on a phishing URL dataset and utilized as a contextual embedding to represent URL tokens, while Bayesian Optimization is employed to identify optimal hyperparameter settings during model training. Experimental results demonstrate that the BERT-LSTM model achieves impressive detection performance, with a precision of 0.9299, recall of 0.9795, F1-score of 0.9540, accuracy of 0.9756, and ROC-AUC of 0.9962. The model consistently outperforms embedding-based methods such as Word2Vec, FastText, and GloVe, as well as a classical baseline model using Logistic Regression with TF-IDF features. These findings suggest that the contextual embeddings generated by BERT effectively capture structural patterns in URLs, leading to more accurate phishing detection and providing a promising approach for enhancing cybersecurity systems.

Keywords: Bayesian Optimization, BERT, Cybersecurity, Deep Learning, Phishing Detection

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

1. INTRODUCTION

The rapid growth of digital technology, particularly the internet and websites, over the past decade has significantly increased the number of phishing sites. These malicious sites attempt to deceive users into providing sensitive information, such as banking credentials and passwords, by impersonating trusted organizations [1]. Phishing attacks commonly exploit malicious links to trick victims, posing serious risks to online banking and e-commerce users [2], [3]. This threat is amplified by the widespread use of smartphones, with 92.6% of the world's 4.66 billion internet users accessing the internet through mobile devices [4]. According to the Anti-Phishing Working Group (APWG), more than 1 million phishing attacks were reported in both the first and second quarters of 2022, marking the highest levels ever recorded and showing a fourfold increase compared to early 2020 [5]. These attacks exploit vulnerabilities on both client and server sides of the uniform resource locator (URL), including compromised servers, user-generated content, and third-party advertisements [6].

Early detection of phishing links is crucial for preventing users from accessing malicious websites that aim to steal personal information. Detection systems identify common characteristics and patterns associated with phishing URLs, enabling proactive measures to be taken before harm occurs [7]. Traditional methods of phishing detection often rely on rule-based approaches that focus on predefined indicators, such as suspicious domain names or unusual URL structures [8]. However, these strategies have become less effective as attackers employ increasingly sophisticated evasion techniques, including domain obfuscation and content manipulation. Modern phishing URLs can closely mimic legitimate links, making them challenging to identify using static rules alone [9]. Consequently, there is a pressing need for more advanced, adaptive detection methods to improve the accuracy and reliability of phishing URL identification.

The advancement of natural language processing (NLP) and machine learning (ML) has created new opportunities for detecting phishing links. Various approaches, including heuristic, rule-based, and ML methods, have been developed to protect users from phishing attacks [10]. However, traditional methods such as blacklist-based detection are often ineffective against zero-day attacks, which exploit previously unknown vulnerabilities and are difficult to mitigate [11]. Conventional ML algorithms such as K-

Nearest Neighbors (KNN), Decision Trees, Random Forests, and Support Vector Machines (SVMs) have been applied to phishing URL detection. Still, their performance depends heavily on effective feature extraction [12], [13], [14]. Deep learning approaches have shown significant improvements by automatically learning complex patterns from data, reducing reliance on manual feature engineering [15]. Models such as Convolutional Neural Networks (CNNs) are effective for feature extraction. At the same time, Recurrent Neural Networks (RNNs), particularly Long Short-Term Memory (LSTM), are better at capturing sequential and contextual relationships in textual data. However, each has its own limitations [16].

Despite advancements in machine learning and deep learning, detecting phishing URLs remains challenging. Many studies still rely on handcrafted features and traditional classifiers, which struggle with complex patterns in obfuscated or dynamically generated URLs. Research on using contextual language models to analyze URL structures is limited, even though URLs can be viewed as sequences of tokens, including domains and paths. This study aims to improve phishing URL detection by combining Bidirectional Encoder Representations from Transformers (BERT) with LSTM networks. By leveraging BERT's contextual embeddings along with LSTM's sequential modeling, the approach seeks to enhance detection performance beyond earlier methods that relied on manual feature extraction.

The primary objective of this research is to integrate BERT with LSTM networks to detect phishing URLs. BERT embeddings provide rich, contextual representations of text that serve as input to the LSTM model. This model is adept at capturing sequential and long-term dependencies within the data, thereby enhancing prediction accuracy. This synergistic approach aims to improve the precision and reliability of phishing URL detection compared to traditional methods. BERT's strength lies in its ability to understand contextual meaning, while LSTM excels in modeling sequence patterns critical for identifying phishing characteristics. Consequently, the combined use of these models could lead to the development of more robust and trustworthy phishing detection techniques for real-world cybersecurity applications.

2. METHODS

This research consists of five stages: data collection, data preprocessing, BERT embedding, model development, and model evaluation.

2.1. Data Collection

At this stage, data collection involved using a Kaggle dataset that contains 514,400 records, organized into two main columns: URL and Label. The URL column includes website addresses, while the Label column classifies each entry into two categories: "bad" and "good." Specifically, the dataset contains 381,359 entries labeled "bad" and 133,041 labeled "good." In this study, the "bad" category is defined as phishing, while the "good" category is non-phishing. This binary classification enables the dataset to be used for phishing URL detection. Figure 1 illustrates the distribution of phishing and non-phishing URLs within the dataset, highlighting the imbalance between the two classes.

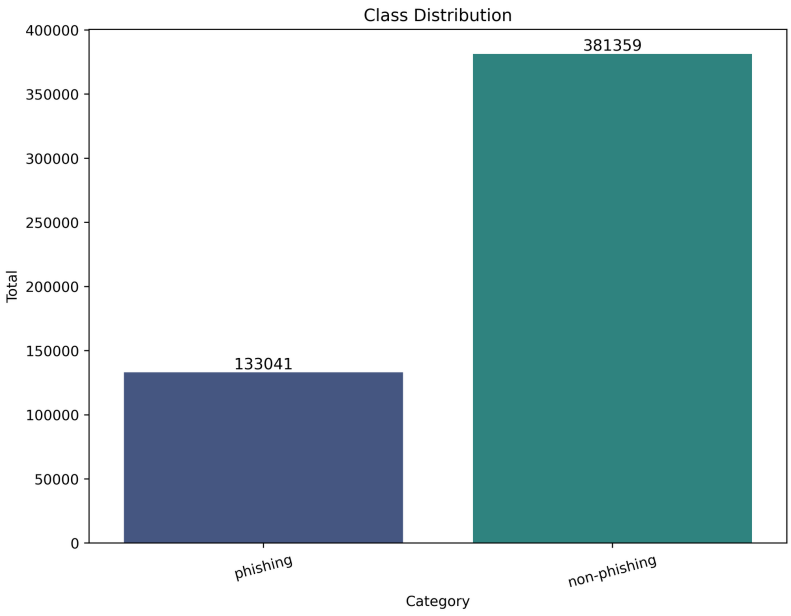

Figure 1. Class Distribution in the Dataset

2.2. Data Pre-processing

The data preprocessing stage is crucial for transforming raw data into a format suitable for model processing. This phase is essential for converting 'dirty' data into 'clean' data, which is then used in the model development. In this study, the data preprocessing process comprises several steps: URL cleaning, tokenization, stemming, text-to-sentence

stage, one-hot encoding, and class imbalance handling. Each of these steps plays a vital role in ensuring that the data is well-prepared for analysis. The detailed preprocessing workflow is illustrated in Figure 2.

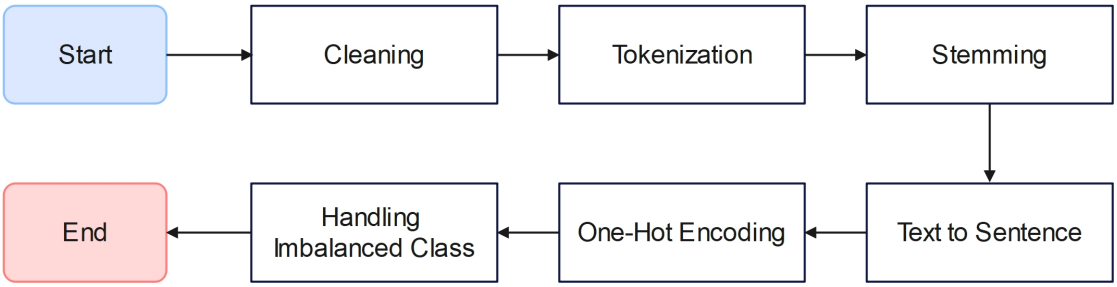

Figure 2. Data Pre-processing Phase

URLs often include structural delimiters, special characters, and irregular token patterns that can introduce noise during model training. As a result, preprocessing is essential to standardize URL structures and extract meaningful textual patterns. The cleaning process involves removing non-informative characters that primarily act as structural separators. Following this, tokenization is performed to break the URL into meaningful segments, such as domain names, directory paths, and file extensions. This allows the model to recognize lexical patterns typically found in phishing URLs. Stemming is then used to reduce token variation and ensure consistency among the tokens. These tokens are converted into sentence-like sequences, enabling the BERT model to interpret the URL as sequential text. Additionally, one-hot encoding is applied to convert categorical class labels into binary vectors suitable for neural network training.

The URL cleaning stage is the initial step in this study, focusing on removing extraneous characters, such as the slash ("/"). Following this, tokenization is applied to split the cleaned URL into individual tokens [17]. Specifically, tokenization is applied to the URLs in the dataset. Next, stemming is performed to obtain the base form of tokens generated during tokenization [18]. The text-to-sentence stage reconstructs the tokens into sentence-like sequences. Subsequently, the one-hot encoding stage transforms the existing classes into binary representations [19]. Table 1 presents an example of the preprocessing results, whereas Table 2 illustrates the application of one-hot encoding.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

Table 1. Example of Pre-processing Result

Phase	Before Pre-processing	After Pre-processing
Cleaning	dieideenwerkstatt.at/css/abc.tar.gz	dieideenwerkstatt at css abc tar gz
Tokenization	dieideenwerkstatt at css abc tar gz	[dieideenwerkstatt, at, css, abc, tar, gz]
Stemming	[dieideenwerkstatt, at, css, abc, tar, gz]	[dieideenwerkstatt, at, css, abc, tar, gz]
Text to sentence	[dieideenwerkstatt, at, css, abc, tar, gz]	dieideenwerkstatt at css abc tar gz

Table 2. One-hot Encoding Representations

Class	One-hot Encoding
non-phishing	[1, 0]
phishing	[0, 1]

The dataset used in this study exhibits a class imbalance, with the number of phishing URLs significantly exceeding that of non-phishing URLs. To address this issue, we employed a class-balancing technique to prevent the model from becoming biased toward the majority class during training. Specifically, we applied random undersampling, which involves selecting a random subset of samples from the majority class to match the minority class's size. This approach creates a more balanced dataset while ensuring that representative samples from both phishing and non-phishing categories are preserved. The distribution of the resulting dataset after applying the class imbalance handling technique is presented in Table 3.

Table 3. Imbalance Class Handling Results

Class	Before Undersampling	After Undersampling
non-phishing	266,951	93,129
phishing	93,129	93,129

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

2.3. BERT Embedding

Word embedding is a technique for converting words into numerical vectors. Traditionally, it is done using techniques such as the bag-of-words model or term frequency-inverse document frequency (TF-IDF). However, nowadays word embeddings can also be generated using more sophisticated methods, such as static and contextual embeddings. Static embedding uses models such as Word2vec and GloVe, which produce fixed vector representations for each word. Meanwhile, contextual embedding involves more complex models such as BERT and Elmo [20]. The model has 12 hidden layers and has been trained using millions of corpus, including BooksCorpus with 800 million words and English Wikipedia with 2,500 million words [21]. BERT produces a word representation of a sentence, using the text-to-sentence from the pre-processing results, as shown in Figure 3.

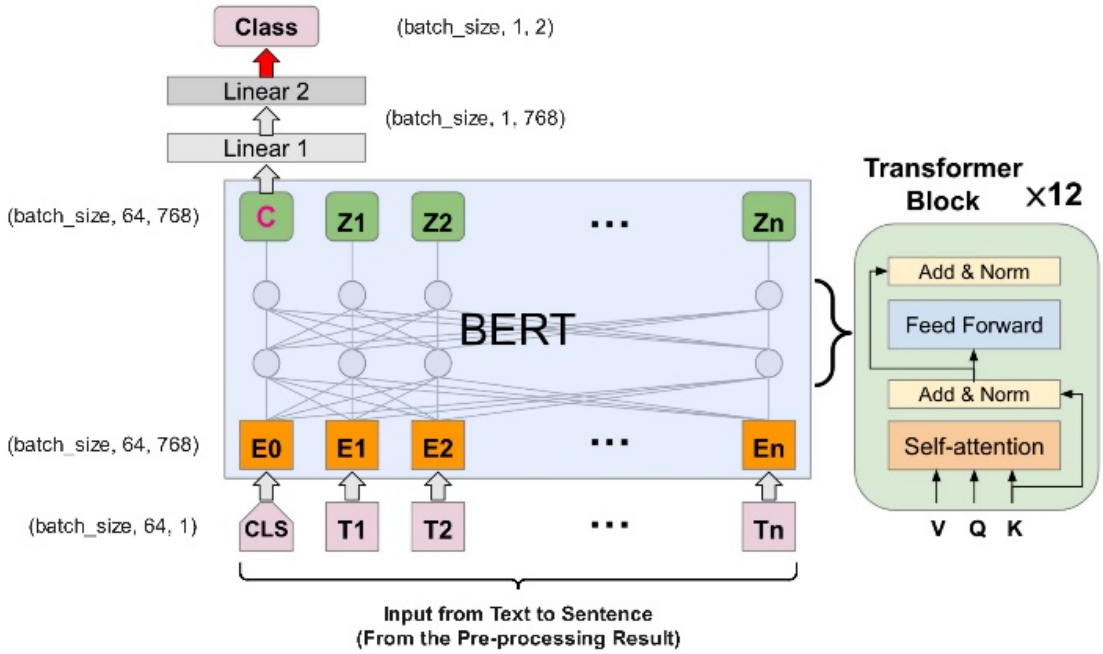

Figure 3. BERT Architecture

In the BERT embedding process, special tokens such as [CLS] and [SEP] are added at the beginning and end of a sentence, respectively, to delineate sentence boundaries. If a word is not present in the BERT vocabulary, it is represented by a hash mark "##." The results of the tokenization process are then converted into token IDs and an Attention Mask. Additionally, the tokenization process transforms sentences into tokens (words)

using the word-piece method, which relies on the model's vocabulary and supplementary vocabulary from external sources. This research uses the "bert-base-uncased" model to transform data into a Tensor format, with 12 transformer layers, 12 attention heads, and an embedding size of 768. The BERT embedding process for phishing detection involves several steps, as illustrated in the diagram below.

1) The input used is "akamai host network com", which is the result of pre-processing from "akamai-hosting-network.com/".
2) Tokenization with WordPiece, resulting in ['aka', '##ma', '##i', 'host', 'network', 'com'].
3) Addition of special tokens [CLS] and [SEP], resulting in ['[CLS]', 'aka', '##ma', '##i', 'host', 'network', 'com', '[SEP]'].
4) Token addition. Padding, or PAD, aligns the lengths of all inputs in a batch. Since the BERT model processes inputs in batches, each input must have the same length. Suppose the maximum length used is 10, it will result in ['[CLS]', 'aka', '##ma', '##i', 'host', 'network', 'com', '[SEP]', '[PAD]', '[PAD]'].
5) Substitute the ID token, which results in [101, 9875, 2863, 2072, 3677, 2897, 4012, 102, 0, 0]. The ID token is a numerical representation of the token or word. For example, CLS is represented as 101, SEP as 102, PAD as 0, and so on.
6) Attention mask which results in [1, 1, 1, 1, 1, 1, 1, 1, 0, 0]. In this final stage, each numerical representation of the vector is treated as an array. Where 1 indicates that the model should attend to the token, while 0 indicates that the token is padding and should be ignored by the model.

In this study, the BERT model is first fine-tuned using the training dataset specific to this research. During this phase, the BERT model's parameters are adjusted to help it adapt to the unique characteristics of the URL data used in the classification task. After the fine-tuning stage, the optimized BERT model is used to generate contextual embeddings for the URL tokens. These embeddings capture the contextual relationships among subword tokens and serve as feature representations for subsequent classification. To ensure reproducibility, a detailed summary of the hyperparameter settings and training configurations used during the BERT fine-tuning stage is presented in Table 4.

Table 4. Configuration on BERT Model Fine-tuning

Components	Description
Pretrained model	bert-base-uncased
Fine-tuning	All BERT parameters are updated
Dropout	0.3
Activation function	Sigmoid
Loss function	Binary cross-entropy
Learning rate	2e-5
Epochs	3
Computing devices	CUDA using NVIDIA T4 GPU

2.4. Model Development

To improve URL phishing detection, a BERT-LSTM approach can be used, combining BERT's contextual representation with LSTM's ability to capture temporal dependencies. BERT is used to extract features from phishing-related URLs or texts. The token representation generated by BERT will capture the contextual information of the URL. Furthermore, LSTM will help capture patterns and temporal dependencies in the URL data that can indicate phishing. Each LSTM layer is added to perform the final classification of whether the URL is phishing.

At this stage, the model architecture of this study is constructed. The architecture comprises several key layers: the input layer, embedding layer, LSTM layer, dropout layer, two fully-connected layers, and the output layer. The output layer will produce predictions of whether the input is non-phishing or phishing. The architecture of the LSTM model is shown in Figure 4. The input layer processes words from sentences, derived from the pre-processing results, specifically, data from the "text to sentence" column. These sentences are then passed to the word embedding layer, which generates vector representations of the words. The output from the embedding layer is fed into the LSTM layer, which processes the data sequentially using time-series methods [22].

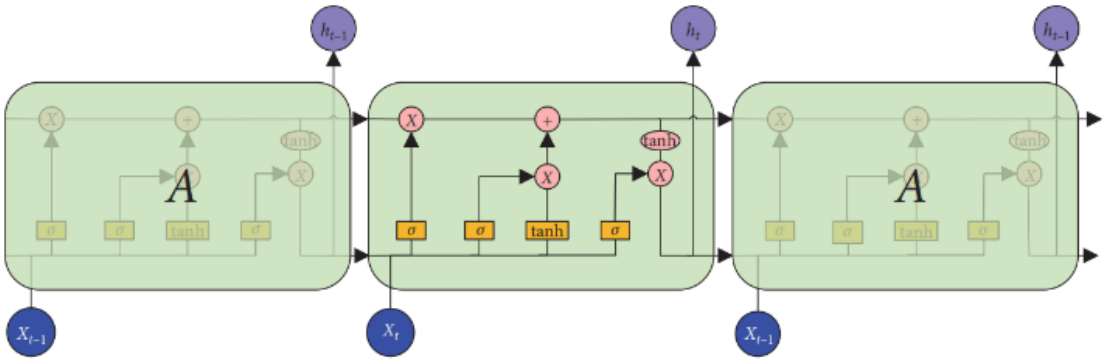

Figure 4. LSTM Architecture

To detect phishing and non-phishing URLs, an LSTM is used to process a sequence of URL data and capture patterns indicative of phishing. This step starts by specifying input data in the form of X_{t-1} and X_t, where $t-1$ and t are input characters or tokens from the URL being parsed. Next, each LSTM cell has three main gates: the forget gate, the input gate, and the output gate. This gate regulates which information is kept or discarded from the cell state. Forget gate decides which information from the previous cell state (C_{t-1}) needs to be forgotten based on the current input (X_t) and the previous hidden state (h_{t-1}). The input gate decides which new information to add to the cell state. It consists of two parts: one using a sigmoid function to determine the updated part (i_t) and one using a tanh function to generate new candidate values ($\widetilde{C}_t$). The cell state is updated by combining information from the forget gate and the input gate. Forget gate sets, which part of the previous cell state should be retained, and the input gate adds new information. Then, the output gate determines which part of the cell state to use as output. The output gate (o_t) generates a value by using a sigmoid function and multiplying it by the updated cell state value. In the final stage, to detect whether a URL is phishing or non-phishing, the last hidden state (h_t) can be used as a feature representation of the URL [23]. This feature representation is then passed to a double fully connected (dense) layer, followed by the Rectified Linear Unit (ReLU) activation function, which generates a probability that the URL is phishing or non-phishing. The dropout layer is applied before the double fully connected layer to reduce overfitting. [24]. This feature representation is then passed to a double fully connected (dense) layer, followed by the ReLU activation function, to generate a probability that the URL is phishing or non-phishing.

In the training process for each model, this study utilized hyperparameter tuning techniques, specifically Bayesian Optimization, to identify the most effective parameter

combinations. This method was designed to enhance the model's ability to learn patterns in the data, resulting in more precise and accurate predictions. The proposed model in this study is an LSTM, while Logistic Regression (LogReg) serves as the baseline for performance comparison. Each model has distinct parameters and varying hyperparameter values, reflecting the unique characteristics of their respective algorithm. Detailed hyperparameter values for each model are shown in Table 5.

Table 5. Hyperparameter Values in Each Model

Models	Parameters	Values
LSTM	Dropout	0.2, 0.4
	Learning Rate	0.0001, 0.001
	Hidden units	64, 128
LogReg	C	0.01, 0.1, 1
	Solver	liblinear, lbfgs

In the proposed architecture, specific parameters are carefully adjusted to improve model performance. The models used in this study are LSTM and LogReg. In the LSTM model, several parameters are hyperparameter-tuned, including the dropout rate, learning rate, and number of hidden units. The dropout rate is set to 0.2 and 0.4 to reduce the risk of overfitting. The learning rate is tested at 0.0001 and 0.001 to control the training speed. Meanwhile, the number of hidden units is set to 64 and 128 to optimize the model's ability to learn word representations. In the Logistic Regression (LogReg) model, two parameters are tuned: the regularization parameter C (0.01, 0.1, and 1) and the solver type (liblinear and lbfgs). Bayesian Optimization is employed to determine the optimal values for these hyperparameters [25], [26]. Table 4 shows the optimized parameters and their values, determined through hyperparameter tuning for each model.

2.5. Model Evaluation

To assess the effectiveness of the proposed model, we use performance metrics derived from the confusion matrix: accuracy, precision, recall, and F1-score. These metrics are calculated using true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). Accuracy measures the overall proportion of correct predictions relative to the total number of predictions, offering a general indication of model performance.

Precision evaluates the proportion of URLs predicted as phishing that are actually phishing, highlighting the model's ability to reduce false positives. Recall assesses the model's effectiveness in correctly identifying actual phishing URLs, thus indicating its sensitivity to detecting threats. The F1-score is the harmonic mean of precision and recall, providing a balanced measure when both false positives and false negatives are significant. The mathematical formulations for accuracy, precision, recall, and F1-score are presented in Formula (1) through Formula (4), respectively.

$$Precision\ (P) = \frac{TP}{TP+FP} \quad (1)$$

$$Recall\ (R) = \frac{TP}{TP+FN} \quad (2)$$

$$F1 - score = 2 * \frac{P * R}{P + R} \quad (3)$$

$$Accuracy = \frac{TP+FN}{TP+FN+TN+FP} \quad (4)$$

3. RESULTS AND DISCUSSION

3.1. Performance Evaluation

3.1.1. Model Optimization

In this stage, model optimization is conducted during training, using a 70/10/20 split of the data for training, validation, and testing, respectively. To improve model performance and achieve higher test accuracy, Bayesian Optimization is used to fine-tune the model's parameters. The parameter registration process begins within a search space that includes dropout, learning rates (0.0001 and 0.001), and LSTM units (64 and 128) with dropout values of 0.2 and 0.4. These parameters are systematically combined via Bayesian Optimization during training, and Table 6 presents the best combinations discovered.

Table 6 compares the performance of the evaluated models, with the BERT-LSTM model emerging as the top performer. Utilizing a dropout rate of 0.2, a learning rate of 0.001, and 256 LSTM units, it achieved an impressive average training accuracy of 0.9963 (99.63%) and a validation accuracy of 0.9753 (97.53%). The minimal gap of about 0.021 between the training and validation accuracies signifies its strong generalization

capability. The FastText-LSTM model closely followed, recording training and validation accuracies of 0.9267 (92.67%) and 0.9483 (94.83%), respectively. In contrast, the Logistic Regression model employing TF-IDF exhibited the lowest performance, underscoring the superiority of BERT-based embeddings integrated with LSTM for effective prediction.

Table 6. Best Parameter Combination During the Training Process

Models	Hyperparameters					Mean Accuracy	
	Dropout	LR	Hidden Unit	C	Solver		
BERT-LSTM	**0.4**	**0.0001**	**64**	**–**	**–**	**Training:**	**0.9963**
						Validation:	**0.9753**
Word2Vec-LSTM	0.2	0.0001	64	–	–	Training:	0.9163
						Validation:	0.9347
FastText-LSTM	0.4	0.0001	64	–	–	Training:	0.9267
						Validation:	0.9483
GloVe-LSTM	0.4	0.001	128	–	–	Training:	0.9193
						Validation:	0.9245
TF-IDF-LogReg	–	–	–	0.01	lbfgs	Training:	0.9049
						Validation:	0.9030

The best parameter combination obtained via Bayesian Optimization yields higher accuracy for the BERT-LSTM model than for the Word2Vec-LSTM, FastText-LSTM, and GloVe-LSTM models. The BERT-LSTM model shows only a small difference between training and validation accuracy, indicating strong generalization and the potential to outperform other models in predictive performance. This superior performance is also influenced by the use of BERT word embeddings, which provide richer contextual word representations than traditional embedding methods such as Word2Vec, FastText, and GloVe, as well as by TF-IDF in the LogReg model. As a result, the BERT-LSTM model is better able to capture semantic relationships within the text data. Therefore, the BERT-LSTM model can be effectively used to predict unseen data in the task of phishing link detection.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

3.1.2. Model Assessment

In this stage, each model is assessed. During the training process, model optimization, namely Bayesian Optimization, is used in the research conducted by L. Zhang et al. [27]. The optimization method produces an optimal model for predicting test data by selecting the best parameter combination, resulting in higher model accuracy. The BERT-LSTM model achieves higher precision, recall, F1 score, and accuracy than other models. This shows that BERT, as a word embedding used in the LSTM model, has advantages in representing words per word from an input, as shown in research by Wicaksana et al. [19]. Figure 5 compares the performance of each model.

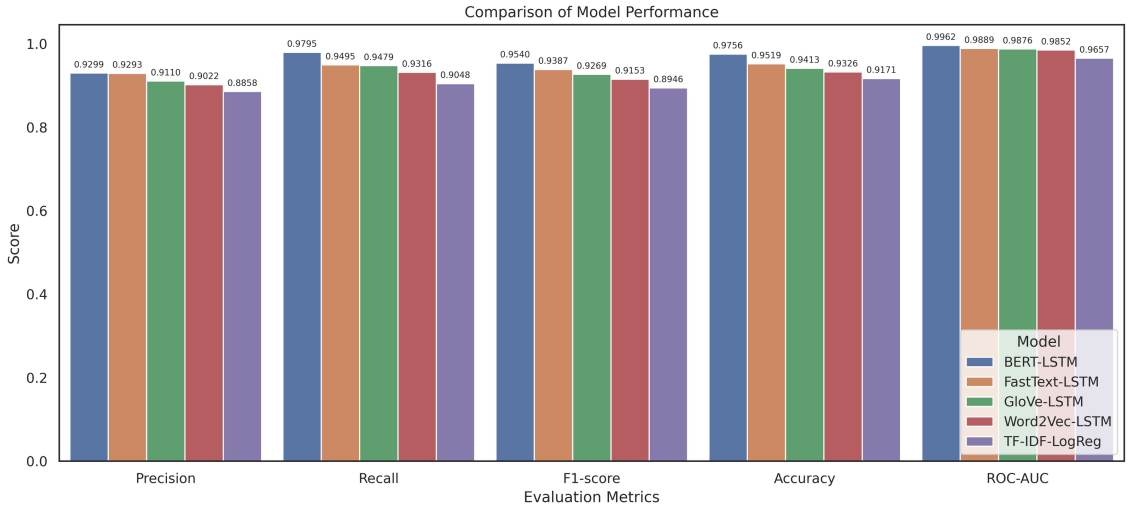

Figure 5. Comparison of Model Performance During the Testing Process

The BERT-LSTM model achieves remarkable performance, as shown in Figure 5, with a precision of 0.9299, a recall of 0.9795, an F1-score of 0.9540, an accuracy of 0.9756, and an ROC-AUC of 0.9962. This model outperforms other frameworks in our study. Our findings show that BERT excels as a word embedding technique for capturing URL vectors, outperforming FastText, Word2Vec, and GloVe. The richer feature extraction is attributed to BERT's training on millions of text samples and its fine-tuning on the URL phishing dataset before being processed by the LSTM model. In contrast, other word embeddings trained on the same dataset do not yield representations as sophisticated as those generated by BERT. Table 7 provides a detailed comparison of precision, recall, F1-score, accuracy, and ROC-AUC across the different models.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

Table 7. Model Performance Assessment

Model	Precision	Recall	F1-score	Accuracy	ROC-AUC
BERT-LSTM	**0.9299**	**0.9795**	**0.9540**	**0.9756**	**0.9962**
FastText-LSTM	0.9293	0.9495	0.9387	0.9519	0.9889
GloVe-LSTM	0.9110	0.9479	0.9269	0.9413	0.9876
Word2Vec-LSTM	0.9022	0.9316	0.9153	0.9326	0.9852
TF-IDF-LogReg	0.8858	0.9048	0.8946	0.9171	0.9657

3.1.3. Model Selection

The analysis of the confusion matrix further confirms that the BERT-LSTM model outperforms all the other evaluated methods. The model effectively classified 74,306 legitimate URLs and 26,062 phishing URLs, demonstrating its strong ability to distinguish between benign and malicious sites. Only 1,966 legitimate URLs were incorrectly classified as phishing, and 546 phishing URLs were misidentified as legitimate, indicating relatively low error rates. The fewest false negatives highlight the model's effectiveness in detecting phishing URLs, a crucial feature for cybersecurity applications. A detailed breakdown of correct and incorrect predictions is shown in Figure 6, which presents the confusion matrix for the selected BERT-LSTM model.

The ROC analysis shows that the proposed BERT-LSTM model performs exceptionally well in distinguishing phishing URLs from legitimate ones. The ROC curve shows a steep rise toward the upper-left corner of the plot, indicating a high True Positive Rate (TPR) while maintaining a very low False Positive Rate (FPR). The model achieves an Area Under the Curve (AUC) value of 0.9962, reflecting its nearly flawless ability to differentiate between the two classes. An AUC value close to 1.0 suggests that the model can effectively identify malicious URLs with minimal misclassification. These findings confirm that the BERT-LSTM model demonstrates highly effective performance in detecting phishing URLs, as illustrated in Figure 7.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

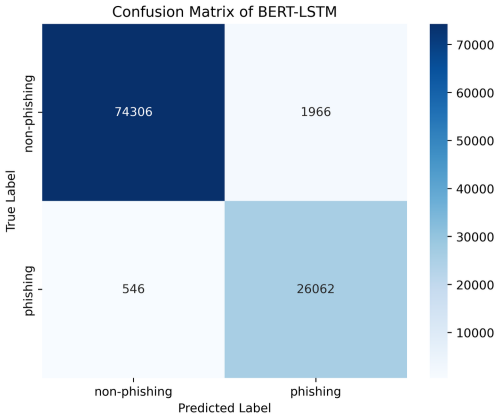

Figure 6. Confusion Matrix of BERT-LSTM

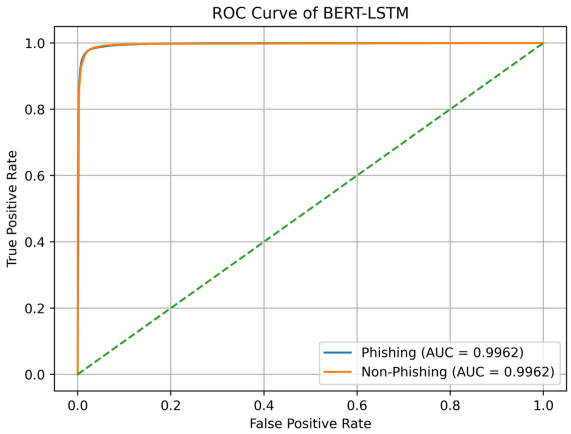

Figure 7. ROC Curve of BERT-LSTM

The BERT-LSTM model presented in this study demonstrates superior performance compared to other models, achieving higher metrics in precision, recall, F1-score, accuracy, and ROC-AUC. These findings underscore its effectiveness in accurately distinguishing between phishing and non-phishing URLs using the provided data. In addition to these quantitative assessments, it is important to examine how each model performs when predicting specific URLs. A detailed analysis of individual predictions can offer deeper insights into how different models classify the same URL instances. Table 8 presents several examples of predictions generated by the evaluated models for selected URL samples.

Table 8 presents a selection of prediction results generated by various models for specific URL samples. This table compares each model's predicted labels with the URLs' actual labels, enabling a direct assessment of the models' performance in specific cases. In many cases, the models produce consistent predictions that align with the actual labels, demonstrating their ability to identify both phishing and non-phishing URLs accurately. However, the table also indicates that some models may produce different predictions for the same URL sample. These examples highlight the behavior of each model when classifying individual URLs in real-world scenarios.

Based on previous studies conducted by Das Guptta et al. [28] and Alzboon et al. [29], each uses machine learning algorithms to detect phishing links. This study employs deep learning algorithms alongside various word embedding techniques as an innovative

approach to differentiate between phishing and non-phishing links. The findings indicate that the BERT-LSTM model achieves superior accuracy. Therefore, the BERT-LSTM model could be effectively utilized in a phishing link detection system, serving as a crucial measure for individuals to remain vigilant against social engineering tactics used by malicious actors attempting to steal personal data.

Table 8. Example of Prediction Results

URL	Model	Actual	Predicted
peaumontreal.com/huds/index.php	BERT-LSTM	phishing	phishing
	FastText-LSTM	phishing	non-phishing
	Word2Vec-LSTM	phishing	phishing
	GloVe-LSTM	phishing	phishing
	Logistic Regression	phishing	phishing
floridamarlinsminors.blogspot.com/2009/02/2-logan-morrison.html	BERT-LSTM	non-phishing	non-phishing
	FastText-LSTM	non-phishing	non-phishing
	Word2Vec-LSTM	non-phishing	non-phishing
	GloVe-LSTM	non-phishing	non-phishing
	Logistic Regression	non-phishing	non-phishing
velavanjewel.com/images/successful/AOL/index-aol.html	BERT-LSTM	phishing	phishing
	FastText-LSTM	phishing	phishing
	Word2Vec-LSTM	phishing	phishing
	GloVe-LSTM	phishing	phishing
	Logistic Regression	phishing	phishing

3.2. Discussion

3.2.1. Effect of BERT Representation on URL Token Patterns

The improved performance of the BERT-LSTM model is primarily due to BERT's ability to generate contextual embeddings that effectively capture the complex token patterns found in URLs. Unlike traditional word embedding methods such as Word2Vec, FastText, and GloVe, which provide static vector representations based solely on local context, BERT creates contextual representations via bidirectional training. This architecture enables BERT to analyze token relationships by considering both preceding and succeeding tokens simultaneously, allowing the model to understand dependencies

throughout the entire URL sequence. As a result, it is better equipped to identify structural patterns in URLs, including suspicious domain structures, abnormal token combinations, and unusual subdomain arrangements commonly associated with phishing attacks. Additionally, BERT employs subword tokenization, enabling it to handle rare or obfuscated tokens commonly found in phishing URLs.

Through subword tokenization, BERT breaks down tokens into smaller, meaningful units. This allows the model to recognize subtle variations in URL components, enabling it to detect suspicious token patterns even when exact tokens are absent from the training vocabulary. As a result, BERT captures both the semantic and structural characteristics of URL sequences, including domain manipulations, inserted keywords, and irregular path structures often found in phishing links. When combined with LSTM's sequential modeling capabilities, the model becomes better at learning dependencies among URL tokens that indicate phishing behavior. Therefore, integrating BERT representations with LSTM greatly enhances the model's ability to identify phishing patterns compared to traditional embedding-based approaches.

3.2.2. Error Analysis of False Positives and False Negatives

The BERT-LSTM model demonstrates impressive performance across various evaluation metrics, though some classification errors remain. As illustrated in Figure 5, the confusion matrix provides a comprehensive overview of the model's classification outcomes. The model successfully identified 74,306 non-phishing URLs and 26,062 phishing URLs, underscoring its strong capability to differentiate between legitimate and malicious links. However, 1,966 non-phishing URLs were incorrectly classified as phishing (false positives), while 546 phishing URLs were misidentified as non-phishing (false negatives). The relatively low number of false negatives indicates the model's effectiveness in detecting phishing URLs, which is essential in cybersecurity contexts, as undetected malicious links can pose significant risks.

False positive cases shown in Figure 5 can occur when legitimate URLs exhibit structural characteristics that resemble phishing patterns, such as multiple nested subdomains, lengthy URL paths, or unusual token sequences. These features may raise suspicions in the detection model, even if the URL is harmless. Conversely, false negatives happen when phishing URLs replicate legitimate domains by using visually similar domain names

or slightly altered tokens that resemble those of trusted websites. These imitation techniques blur the lines between phishing and legitimate URLs, making detection more challenging. These observations highlight the difficulties of phishing detection, especially as attackers continuously adapt URL structures to evade automated detection systems.

3.2.3. Computational Cost versus Performance Trade-off

The BERT-LSTM model demonstrates outstanding predictive performance but requires significantly more computational resources during training than other models. In this study, fine-tuning the BERT component took 7,021.61 seconds (approximately 1.95 hours), with peak GPU memory usage reaching 2.47 GB. This level of resource consumption surpasses that of models using traditional embedding techniques such as Word2Vec, FastText, and GloVe, as well as the classic TF-IDF used in the logistic regression baseline model. The increased resource requirements are primarily due to the large number of parameters in the BERT architecture and the additional sequence modeling performed by the LSTM layer. As a result, training the BERT-LSTM model is more computationally intensive than the other approaches evaluated.

Despite its higher computational cost, the BERT-LSTM model outperforms all other evaluated models across multiple evaluation metrics. As shown in Table 6, the BERT-LSTM model achieves a precision of 0.9299 (92.99%), a recall of 0.9795 (97.95%), an F1-score of 0.9540 (95.40%), an accuracy of 0.9756 (97.56%), and a ROC-AUC of 0.9962 (99.62%). These results demonstrate that the contextual representation capabilities of BERT significantly enhance the model's ability to identify phishing URLs. The superior predictive performance indicates that the additional computational cost associated with the BERT fine-tuning process is justified. Therefore, the BERT-LSTM model effectively balances computational complexity and detection performance for phishing URL classification tasks.

3.2.4. Limitations of the Study

While the proposed BERT-LSTM model shows strong performance, this study acknowledges several limitations. One significant limitation concerns the training process, particularly during the BERT fine-tuning phase before it is used as a contextual embedding for the LSTM architecture. This fine-tuning phase requires additional training time, as the pre-trained BERT model's parameters must be adjusted to fit the phishing

URL dataset used in this research. Consequently, this extra training step increases computational complexity compared to models that rely on traditional embedding techniques without pre-trained language models. As a result, the training process becomes more resource-intensive, potentially reducing efficiency, especially when the model requires frequent retraining or updates.

One limitation of this study concerns the characteristics of the dataset used. The dataset employed for training and evaluation is not fully balanced between phishing and non-phishing URLs. This imbalance may affect the classification model's learning process. Additionally, the model's performance heavily relies on the dataset it was trained on, which means the patterns it learns may reflect the specific traits of that data. This reliance can hinder the model's ability to generalize to new phishing tactics or previously unseen URL patterns. Therefore, future research should focus on using more diverse datasets and implementing class-balancing techniques to enhance the robustness and generalization capabilities of phishing detection models.

4. CONCLUSION

This study introduces a BERT-LSTM-based approach designed to improve the detection of phishing URLs. During the training phase, Bayesian Optimization was utilized to determine the optimal hyperparameter configuration for the BERT-LSTM model. The best combination identified through this optimization process included a dropout rate of 0.4, a learning rate of 0.0001, and 64 hidden units for the LSTM. With this configuration, the proposed model achieved impressive results: precision of 0.9299, recall of 0.9795, F1-score of 0.9540, accuracy of 0.9756, and ROC-AUC of 0.9962. These results outperformed those of other comparative models, including Word2Vec-LSTM, FastText-LSTM, GloVe-LSTM, and Logistic Regression. These findings suggest that integrating BERT contextual embeddings with LSTM's sequential modeling effectively captures complex patterns in phishing URLs. While the results are promising, several limitations should be considered. The model was trained and evaluated on a specific dataset, so its characteristics may influence the patterns it learned. Furthermore, the evaluation did not include external datasets or zero-day phishing scenarios, potentially limiting the model's ability to generalize to new phishing techniques. The BERT-LSTM architecture also requires significant computational resources due to the fine-tuning process needed for the BERT

model before it can be used as a contextual embedding for the LSTM layer. Future research could focus on evaluating the model across different datasets, detecting zero-day phishing attacks, and developing more computationally efficient architectures to improve the practical applicability of phishing detection systems.

REFERENCES

[1] A. Aljofey, Q. Jiang, A. Rasool, H. Chen, W. Liu, Q. Qu, and Y. Wang, "An effective detection approach for phishing websites using URL and HTML features," *Sci. Rep.*, vol. 12, no. 1, p. 8842, May 2022, doi: 10.1038/s41598-022-10841-5.

[2] R. Ahmad, S. Terzis, and K. Renaud, "Getting users to click: a content analysis of phishers' tactics and techniques in mobile instant messaging phishing," *Inf. Comput. Secur.*, vol. 32, no. 4, pp. 420–435, Sep. 2024, doi: 10.1108/ICS-11-2023-0206.

[3] S. Vinoth, H. L. Vemula, B. Haralayya, P. Mamgain, M. F. Hasan, and M. Naved, "Application of cloud computing in banking and e-commerce and related security threats," *Mater. Today Proc.*, vol. 51, pp. 2172–2175, 2022, doi: 10.1016/j.matpr.2021.11.121.

[4] M. Miao, T. Jalees, S. I. Zaman, S. Khan, N.-A. Hanif, and M. K. Javed, "The influence of e-customer satisfaction, e-trust and perceived value on consumer's repurchase intention in B2C e-commerce segment," *Asia Pac. J. Mark. Logist.*, vol. 34, no. 10, pp. 2184–2206, Nov. 2022, doi: 10.1108/APJML-03-2021-0221.

[5] A. C. Tally, J. Abbott, A. M. Bochner, S. Das, and C. Nippert-Eng, "Tips, tricks, and training: Supporting anti-phishing awareness among mid-career office workers based on employees' current practices," in *Proc. CHI Conf. Hum. Factors Comput. Syst.*, 2023, pp. 1–13.

[6] M. Nanda, M. Saraswat, and P. K. Sharma, "Enhancing cybersecurity: A review and comparative analysis of convolutional neural network approaches for detecting URL-based phishing attacks," *e-Prime Adv. Electr. Eng. Electron. Energy*, vol. 8, no. November 2023, p. 100533, 2024, doi: 10.1016/j.prime.2024.100533.

[7] S. Kavya and D. Sumathi, "Staying ahead of phishers: a review of recent advances and emerging methodologies in phishing detection," *Artif. Intell. Rev.*, vol. 58, no. 2, p. 50, Dec. 2024, doi: 10.1007/s10462-024-11055-z.

[8] A. S. Noviantina and F. A. Yulianto, "A Hybrid Approach for Detecting Phishing URLs: Integrating Rule-Based and Machine Learning Techniques," in *Proc. Int. Conf. Inf. Commun. Technol. (ICoICT)*, IEEE, Jul. 2025, pp. 1–6. doi: 10.1109/ICoICT66265.2025.11193051.

[9] W. Li, S. Manickam, Y.-W. Chong, W. Leng, and P. Nanda, "A State-of-the-Art Review on Phishing Website Detection Techniques," *IEEE Access*, vol. 12, pp. 187976–188012, 2024, doi: 10.1109/ACCESS.2024.3514972.

[10] A. Jadhav and P. Chandre, "A Hybrid Heuristic-Machine Learning Framework for Phishing Detection Using Multi-Domain Feature Analysis," *Eng. Technol. Appl. Sci. Res.*, vol. 15, no. 5, pp. 27219–27226, Oct. 2025, doi: 10.48084/etasr.11548.

[11] F. Carroll, J. A. Adejobi, and R. Montasari, "How good are we at detecting a phishing attack? Investigating the evolving phishing attack email and why it continues to successfully deceive society," *SN Comput. Sci.*, vol. 3, no. 2, p. 170, 2022.

[12] M. Irsan, F. Febriana, H. H. Nuha, and H. R. Putra Sailellah, "Phishing Detection on URL Data Using K-Nearest Neighbors Method," in *Proc. Int. Conf. Innov. Intell. Informatics Comput. Technol. (3ICT)*, IEEE, Nov. 2024, pp. 792–797. doi: 10.1109/3ict64318.2024.10824630.

[13] A. Karim, M. Shahroz, K. Mustofa, S. B. Belhaouari, and S. R. K. Joga, "Phishing Detection System Through Hybrid Machine Learning Based on URL," *IEEE Access*, vol. 11, pp. 36805–36822, 2023, doi: 10.1109/ACCESS.2023.3252366.

[14] H. S. Wicaksana and K. Huda, "Penerapan Word2Vec dan SVM dengan Hyperparameter Tuning untuk Deteksi Phishing," *JURIKOM (Jurnal Riset Komputer)*, vol. 12, no. 3, pp. 361–371, Jun. 2025, doi: 10.30865/jurikom.v12i3.8729.

[15] Z. Alshingiti, R. Alaqel, J. Al-Muhtadi, Q. E. U. Haq, K. Saleem, and M. H. Faheem, "A Deep Learning-Based Phishing Detection System Using CNN, LSTM, and LSTM-CNN," *Electronics*, vol. 12, no. 1, p. 232, Jan. 2023, doi: 10.3390/electronics12010232.

[16] E. A. Aldakheel, M. Zakariah, G. A. Gashgari, F. A. Almarshad, and A. I. A. Alzahrani, "A Deep learning-based innovative technique for phishing detection in modern security with uniform resource locators," *Sensors*, vol. 23, no. 9, p. 4403, 2023.

[17] P. E. Shawky, S. M. ElKaffas, and S. K. Guirguis, "Effect of typos on text classification accuracy in word and character tokenization," *J. Adv. Res. Appl. Sci. Eng. Technol.*, vol. 40, no. 2, pp. 152–162, 2024.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

[18] A. Louati, H. Louati, E. Kariri, F. Alaskar, and A. Alotaibi, "Sentiment analysis of Arabic course reviews of a Saudi university using support vector machine," *Appl. Sci.*, vol. 13, no. 23, p. 12539, 2023.

[19] H. S. Wicaksana, R. Kusumaningrum, and R. Gernowo, "Determining community happiness index with transformers and attention-based deep learning," *IAES Int. J. Artif. Intell.*, vol. 13, no. 2, pp. 1753–1761, Jun. 2024, doi: 10.11591/ijai.v13.i2.pp1753-1761.

[20] D. S. Asudani, N. K. Nagwani, and P. Singh, "Impact of word embedding models on text analytics in deep learning environment: a review," *Artif. Intell. Rev.*, vol. 56, no. 9, pp. 10345–10425, Sep. 2023, doi: 10.1007/s10462-023-10419-1.

[21] M. Wankhade and A. C. S. Rao, "Opinion analysis and aspect understanding during covid-19 pandemic using BERT-Bi-LSTM ensemble method," *Sci. Rep.*, vol. 12, no. 1, p. 17095, Oct. 2022, doi: 10.1038/s41598-022-21604-7.

[22] K. L. Tan, C. P. Lee, K. S. M. Anbananthen, and K. M. Lim, "RoBERTa-LSTM: A Hybrid Model for Sentiment Analysis With Transformer and Recurrent Neural Network," *IEEE Access*, vol. 10, pp. 21517–21525, 2022, doi: 10.1109/ACCESS.2022.3152828.

[23] A. Ozcan, C. Catal, E. Donmez, and B. Senturk, "A hybrid DNN–LSTM model for detecting phishing URLs," *Neural Comput. Appl.*, vol. 35, no. 7, pp. 4957–4973, Mar. 2023, doi: 10.1007/s00521-021-06401-z.

[24] H. Li, G. K. Rajbahadur, D. Lin, C.-P. Bezemer, and Z. M. Jiang, "Keeping Deep Learning Models in Check: A History-Based Approach to Mitigate Overfitting," *IEEE Access*, vol. 12, pp. 70676–70689, 2024, doi: 10.1109/ACCESS.2024.3402543.

[25] H. S. Wicaksana and K. Huda, "Optimized Machine Learning Approach for Malware Detection using Bayesian Optimization," *J. Sisfokom (Sist. Inf. dan Komput.)*, vol. 15, no. 01, pp. 103–111, Dec. 2025, doi: 10.32736/sisfokom.v15i01.2547.

[26] K. Yang, L. Liu, and Y. Wen, "The impact of Bayesian optimization on feature selection," *Sci. Rep.*, vol. 14, no. 1, Dec. 2024, doi: 10.1038/s41598-024-54515-w.

[27] L. Zhang *et al.*, "CNN-LSTM Model Optimized by Bayesian Optimization for Predicting Single-Well Production in Water Flooding Reservoir," *Geofluids*, vol. 2023, 2023, doi: 10.1155/2023/5467956.

[28] S. Das Guptta, K. T. Shahriar, H. Alqahtani, D. Alsalman, and I. H. Sarker, "Modeling Hybrid Feature-Based Phishing Websites Detection Using Machine Learning Techniques," *Ann. Data Sci.*, vol. 11, no. 1, pp. 217–242, Feb. 2024, doi: 10.1007/s40745-022-00379-8.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

[29] M. S. Alzboon, M. Subhi Al-Batah, M. Alqaraleh, F. Alzboon, and L. Alzboon, "Phishing Website Detection Using Machine Learning," *Gamification Augmented Real.*, vol. 3, p. 81, Jan. 2025, doi: 10.56294/gr202581.