DEEP LEARNING APPROACH FOR DETECTION OF PHISHING ATTACK TO STRENGTHEN NETWORK SECURITY

Hadeer Alsubaie, Rahaf Althomali and Samah Alajmani

Department of Information Technology, College of Computer and Information Technology, Taif University, PO Box. 11099, Taif 21994, Saudi Arabia

ABSTRACT

Phishing attacks are one of the most aggressive vulnerabilities in cybersecurity networks, typically carried out through social engineering and URL obfuscation. Traditional detection methods struggle to combat advanced techniques applied. In this paper, a deep learning-based approach is proposed to increase the accuracy of phishing detection while reducing the number of false positives. Four models: CNN-BLSTM, SNN, Transformer, and DBN, are developed and evaluated on a phishing dataset that includes critical features such as URL structure, domain age, and presence of HTTPS. The other model, CNN-BLSTM, achieved 98.9% better accuracy, effectively linking URL sequences in space and time. It is found that although deep learning models have a significant improvement over traditional methods in detecting phishing attacks, the level of computational resources still prevents them from real-time applications. Further research includes hybrid models and adversarial approaches to improve state-ofthe-art and practical solutions to address phishing threats. This study highlights a new technological application to Internet security concerns, particularly in the area of combating phishing.

KEYWORDS

Network Cybersecurity, Phishing Detection, URL, Web security, Deep Learning

1. INTRODUCTION

The Internet has emerged as the basic structure for conducting and managing essential operations in modern day life [1]. Industries dependent on financial transactions, such as banking and e-commerce, rely significantly on the Internet for efficient service delivery [2]. The digital connectivity provides significant advantages but also presents several security challenges for enterprises and individuals [3]. Phishing attacks have surfaced as a considerable hazard among these difficulties. Phishing is a type of identity theft that leverages human vulnerabilities via social engineering and psychological manipulation, allowing attackers to mislead users into disclosing important information. Phishing attacks come in various forms, such as Website Phishing, Phishing through Online Social Networks [4], Email Phishing [5], SMS Phishing [6], and Voice Phishing [7]. The ultimate goal of phishing is to extract personal data, including usernames and passwords, credit card numbers, and other useful information that can be converted into money or other illicit gains. These attacks generally entail the impersonation of credible entities, such as financial institutions or esteemed corporations. It uses misleading emails, text messages, or counterfeit websites [8]. The purloined information is subsequently exploited for financial fraud or other unlawful objectives, resulting in significant harm to both persons and organizations.

Although advancements in cybersecurity solutions and rigorous management of network infrastructures, phishing attacks remain a significant issue. Cybercriminals perpetually adapt their strategies, exploiting the evolving characteristics of digital platforms to augment their deceitful prowess [9]. URL phishing, a particularly nefarious variant of phishing, entails enticing people to counterfeit websites through misleading connections [10]. As these attacks grow more sophisticated, conventional detection measures fail to keep up, rendering people susceptible to exploitation. The financial and reputational damages resulting from phishing underscore the pressing necessity for effective remedies. This study seeks to address the increasing escalating of phishing attacks by utilizing new technologies, such as deep learning and complex analytics, to improve detection and prevention. The main goal of the research is to make use of deep learning algorithms and data-driven techniques in developing more efficient and effective solutions for phishing detection.

The study contribution can be summarized as follows:

1. Develops deep learning algorithms for protecting against complex phishing attacks, improving cybersecurity.
2. Assists prevent phishing risks with real-time detection and reaction.
3. Discusses different deep learning models and evaluate results using performance metrics to reduce false positives for improving system reliability and user confidence.
4. Hybrid and advanced deep learning methods can improve phishing detection and identify new assaults.

2. RELATED WORKS

Phishing attacks are one of the common methods of cybercrime, which mainly rely on deception and social engineering to extract information from people, especially high-value information that can be used for fraud or identity theft purposes. Most phishing attacks imitate real businesses - banks and online services. Phishing websites are the most common type of phishing attack, in which fraudulent websites are created copying some authentic ones to extract personal information from users, such as user names and passwords. Most of these sites are designed to look like the actual login pages for big services (PayPal, Google) [11]. Moreover, studies have looked at the effects of such attacks on user behavior and ways to increase awareness in terms of encouraging users to check URLs before clicking. Only in the recent past have researchers started to focus on using artificial intelligence and other technologies to increase the efficiency of security systems in combating phishing attacks, with a particular focus on the increased accuracy and speed of detecting malicious URLs in real time.

Sahingoz et al. [12], presented a deep learning-based system that uses five distinct architectures artificial neural networks (ANNs), convolutional neural networks (CNNs), recurrent neural networks (RNNs), bidirectional recurrent neural networks (BRNNs), and attention networks to detect phishing attempts. The system evaluated URLs by embedding characters, allowing it to operate independently of the language in the URLs. compiled a significant, well-balanced dataset of over 5.1 million URLs, which includes 2.32 million phishing URLs supplied from PhishTank and 2.88 million legitimate URLs acquired using Common Crawl. Convolutional neural networks attained a maximum accuracy of 98.74% in detecting phishing attempts and demonstrated effective performance against zero-day attacks. Nevertheless, the system had shortcomings in handling complex attacks like URL hijacking. Remya [13], devised a successful phishing detection technique utilizing convolutional neural networks (CNN) in conjunction with residual pipelining. It focuses the analysis of URLs to discern characteristics that aid the model in recognizing them as either authentic or phishing. The research employed a Kaggle dataset including 651,191 URLs, which included 94,111 phishing URLs and 428,103 legal URLs. The

model exhibited exceptional performance, attaining an accuracy of 98.29%, proving its efficacy in identifying phishing attacks. But a constraint was observed concerning the requirement for significant processing resources owing to the difficulty of the deep learning models. Sawant et al. [14], presented a powerful hybrid phishing detection model that combines machine learning and deep learning techniques. They used two Kaggle datasets: one with 11,430 URLs and 87 features, and another with 651,191 URLs and two features. The model combines random forests with CNNs and takes advantage of the strengths of both techniques. The hybrid model achieved 97% accuracy, topping individual models. However, a significant deficiency is the lack of testing against sophisticated threats such as GANs. Nishitha et al. [15], Analyzed phishing URL detection using many machine learning and deep learning methods. The models used were Logistic Regression, KNN, Decision Tree, Random Forest, SVM, and CNN, amongst which were tested for their performances using accuracy and efficiency. A dataset that entails 549,346 URLs from different sources was used; 5,000 instances were chosen for training, which embodied 75% of legitimate URLs and 25% of phishing. Preparing strategies as Regex Tokenizer, Snowball Stemmer, and Count Vectorizer were employed to improve the models' accuracy. CNN had outstanding results, achieving a 96% accuracy rate, followed by logistic regression at 94.33%. Nevertheless, the study did not look into hybrid models, which potentially improved their efficiency. Kaushik et al. [16], Proposed a deep learning solution for phishing attack detection by deploying the CNN model (LSTM). It works on features extracted from URL and email content for phishing detection. Experiments were conducted with a real dataset (collected) containing phishing emails and legitimate emails by using a hybrid model (CNN-LSTM). Finally, they extracted local patterns from the text and then use CNN to build features for the time series solution with the input as embedding vectors for CNN-LSTM. The only limitation here is new bug and attack data. Ozcan et al. [17], presented a hybrid model of DNNs and LSTMs for link phishing identification. The two primary datasets used were Ebbu2017 (legitimate links, 36,400 and phishing links, 37,175) and PhishTank (26,000 links). This hybrid model can leverage both manually extracted and machine-generated features and that us why they have significant performance improvements. According to the results, him model performs 98.19% better than others in detecting phishing links. The gap here lies in the challenges related to the high complexity of phishing detection models due to the diversity of feature extraction techniques and the different quality of data used. Bozkir et al. [18] introduced a new model to detect phishing websites using n-gram features that can be computed without pre-training or manual feature engineering. Hierarchically stacked network layer model consisting of CNN, LSTM, and attention process They got good and fast at finding patterns in links by selecting n-grams in a smart way. They trained the model on a new dataset of 800,000 URLs (400,000 phishing and 400,000 legitimate). The model is built for highest performance and real-time processing and does not rely on the use of libraries or external services. This model could provide an accuracy of 98.27%, better than other models that worked under adversarial attack. The most prominent gap is the non-diversarity of previous datasets and their dependence on handcrafted features, which were prone to fail when integrated in real-world scenarios. Benavides-Astudillo et al. [19], Presented a model for mining text from web pages to automatically detect phishing attacks by deep learning and natural language processing: The model will use DL and NLP techniques to mine text from web pages and automatically detect phishing attacks. The GloVe model will be incorporated using the Keras Embedding API to obtain the textual features of semantics and syntax. for executing the neural network, such algorithms as LSTM, BiLSTM (2), GRU, and BIGRU were used, with BiGRU giving constant results having an accuracy rate of 97.39%. They have experimented on the Phishload dataset that contains 10,373 samples of phishing pages and legitimate pages. "It actually detects embedded links—but not in the classical sense (URL links)—in those texts. Very few studies analyze text using NLP techniques where other advanced models like attention mechanisms help improve the performance. Table 2 summarizes recent phishing detection studies.

Table 2. Summary of recent phishing detection studies.

Ref.	Best Model	Datasets	Acc	Strengths	Weaknesses
[12] (2024)	CNN	PhishTank / Common Crawl	98.74%	Ability to detect Zero-Day attacks.	Requirement for significant computational resources.
[13] (2024)	Residual Pipelining - CNN	-	98.29%	Achieving 98.29% accuracy in phishing detection using deep learning techniques.	
[14] (2024)	Hybrid Model - RF, CNN	Kaggle	97%	The hybrid model achieved 97% accuracy.	It was not tested against advanced GANs attacks.
[15] (2023)	CNN	Kaggle	96%	CNN with 96% accuracy.	No use of hybrid models
[16] (2023)	CNN and LSTM	Real-world dataset	High	Using hybrid models for text analysis and attack detection.	New attack data might pose a challenge.
[17] (2023)	DNN and LSTM	Ebbu2017, PhishTank	98.19%	High performance in detecting phishing links using extracted features.	The complexity of phishing detection models due to diverse feature extraction techniques and data quality.
[18] (2023)	CNN, LSTM, and Attention Process	800,000 URLs (Phishing + Legitimate)	98.27%	High real-time performance.	Heavily relies on handcrafted features, which might fail in real-world scenarios.
[19] (2023)	BiGRU and GloVe	Phishload	97.39%	Uses NLP techniques to extract text and analyze embedded links.	Few studies use advanced NLP models such as attention mechanisms to improve performance.

3. METHODOLOGY

This section outlines the methodology used to select the study's methods, design, and analytical techniques. The research was conducted through several critical steps detailed in the following methodology. The first step involved identifying the research problem, which focused on detecting phishing attacks through deep learning algorithms. This issue was contextualized within the contemporary challenges of phishing detection and its impact on cybersecurity. Next, a review of related work concerning phishing detection and deep. The Phishing Attack Dataset was chosen due to its diverse phishing URLs, which is crucial for training robust models capable of detecting various phishing techniques. The algorithms that will be experimented with include Transformers [20], Deep Belief Networks (DBNs) [21], Spiking Neural Networks (SNNs) [22], Bidirectional Long Short-Term Memory (BLSTM) networks [23], and Convolutional Neural Networks (CNNs) as shown in Figure 1. The fourth step involves studying the architecture of the proposed algorithms and implementing them using the Phishing Attack Dataset to evaluate their performance [24]. The models will be tested to identify which achieves the highest accuracy in detecting phishing attacks, focusing on the effectiveness of each model regardless of the number of features in the dataset. The fifth step is to develop the model architecture and conduct

experiments with the selected deep-learning algorithms. Based on the nature of the problem and the dataset, the best-performing deep learning model will be selected. This model will be implemented and trained on the Google Colab platform [25], where hyperparameter tuning will be conducted to optimize performance. Finally, the model's performance will be evaluated using standard metrics, including accuracy, precision, recall, and F-score. Further improvements will be made through an iterative optimization process, continuously refining the model based on results from previous studies to ensure ongoing enhancement.

Figure 1. The Overall Research Methodology.

With the growing of phishing attacks, the urgency of developing an effective detection system to protect users has never been greater. We introduce advanced deep learning model for detecting phishing attacks using deep learning techniques. It outlines the methodologies, tools, and steps for building this model. Additionally, the approach for training and testing the model is detailed to ensure high accuracy in identifying phishing URLs. For this study, the Phishing Attack Dataset obtained from IEEE Data Port was utilized, comprising 11,504 records and 32 URL characteristic features related to factors such as HTTPS presence and domain attributes. The data underwent pre-processing to ensure quality and consistency, followed by applying feature selection methods to eliminate irrelevant and redundant features. The dataset was divided into 70% for training, 20% for validation, and 10% for testing. Deep learning models, including Transformer, DBN, SNNs, BLSTM, and CNNs, were trained on the split data. These models were specifically designed to learn complex patterns associated with phishing URLs, which traditional approaches may find challenging to address. Deep learning enhances accuracy and efficiency in phishing detection, reducing false positives and enabling the early identification of more sophisticated phishing URLs. The phishing detection process involves data collection, pre-processing (such as handling missing data, encoding, and standardization), feature selection, and the subsequent dataset splitting into training, validation, and test sets. Deep learning models, including Transformer, DBN, SNNs, BLSTM, and CNNs, were trained to classify URLs into safe and unsafe categories. Figure 2 illustrates the complete workflow of this process.

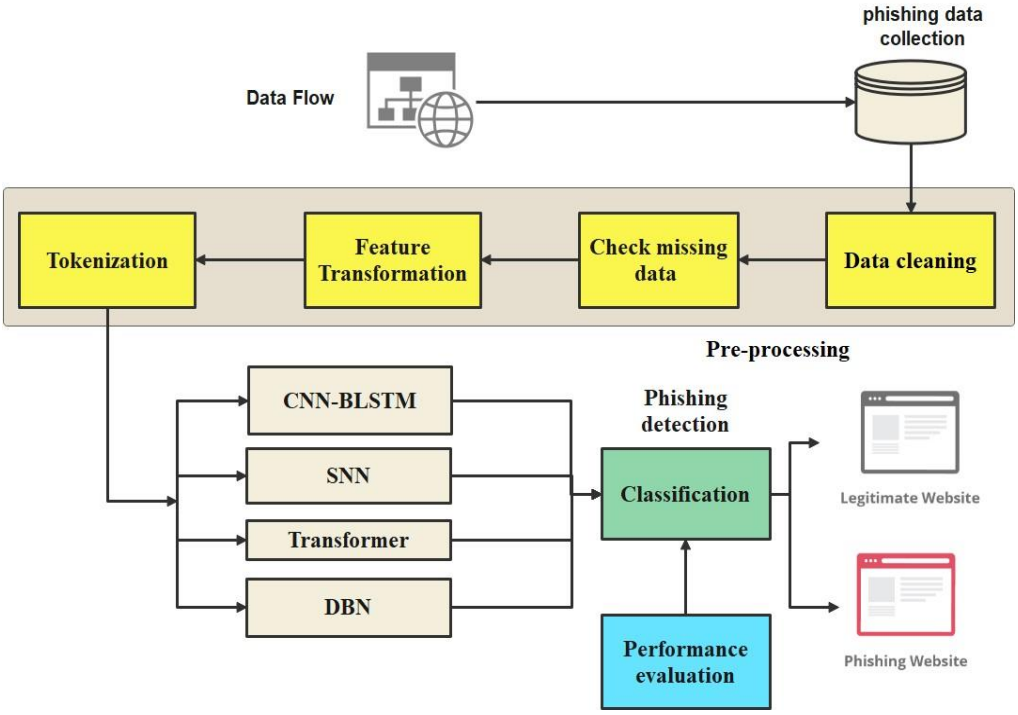

Figure 2. The Proposed deep learning model

3.1. Phishing Attack Dataset

The Phishing Attack Dataset [26] is most appropriate since it best reflects a real-world distribution of URLs (both phishing and legitimate). It is a well-labeled dataset coming with thousands of URLs belonging to these two major categories legitimate and phishing. This gives enough diversity and volume in data for training, testing, and validating the models of detecting phishing, including advanced machine learning and deep learning algorithms. The dataset used consists of various URL features; length of the URL, URL structure, domain properties, and other important characteristics usually considered common factors in executing a phishing attack. Every URL in the dataset was marked against its class, that is either phishing or legitimate, this proved to be beneficial during the course of supervised learning. Real datasets are made up of diverse datasets because the URLs are taken from diversified sectors like finance, e-commerce, and social media, which represent characteristic dimensions into which fraudulent attacks around phishing revolve.

Moreover, since the dataset is in structured CSV form, preprocessing and integrating into any deep learning framework become very easy. Dataset has each of its rows corresponding to a URL and columns with various features related to URL length, some special characters, domain properties, etc. which are important in learning the patterns of phishing URLs and hence building strong detection models. The Phishing Attack Dataset is one of the freely available datasets in IEEE DataPort for research purposes. Since it is updated at very short intervals, one should expect that models learned on this data will show very good detecting performance of both current and novel phishing techniques. The dataset is friendly to a variety of machinelearning and deep-learning algorithms. It includes Transformer, DBN, SNNs, BLSTM, CNNs among others that can be used to increase accuracy in detection. The richness of the features and URL diversities make it ideal for developing models that would effectively fight phishing attacks in real environments of use. This dataset would allow different researchers to test multiple algorithms developing

systems for detecting phishing. Datasets are shared under dataset/ folder. The dataset structure as shown in Table 3, has the following structure.

Table 3. Detailing about the phishing attacks dataset.

Num	component	Details
1	Dataset Versions	Two versions: Small Dataset (10% sample) and Big Dataset (full dataset).
2	Sources	Phishing URLs from Phishtank and legitimate URLs from CommonCrawl.
3	Files in Both Versions	- meta.txt: Metadata - train.txt: 70% training data - test.txt: 10% testing data - val.txt: 20% validation data
4	Additional	- top100k_cc.txt: List of top 100,000 legitimate domains. - top_100k_errored_cc.txt: Domains with parsing errors.
5	Phishing URL Count	Approximately 114,000 phishing URLs collected.
6	Legitimate	Approximately 100,000 legitimate URLs included.

train.txt: This file contains the training data, representing about 70% of the total data available and which would be used to train the model. test.txt: This file contains the test data, representing about 10% of the total data available and which would be used to evaluate model performance after training. val.txt: This file contains the validation data, representing about 20% of the total data available and which would be used in model fine-tuning during training (to prevent overfitting).

3.2. Pre-Processing

Pre-processing is a very important element in deep learning models since it has the capacity to play a foundational role in increasing the classifier's performance and increasing the overall accuracy of classification. In the preprocessing phase of this study, first, the types of features available in the dataset are identified, like numeric and non-numeric types of data. Some of these features are very important with regard to the detection of phishing, while others may be redundant and bring unnecessary noise leading to degradation of both speed and accuracy of the training process. By taking the unnecessary features off first, we can now proceed to the next stages. In the work, a simple preprocessing step had been used where the hexadecimal values stored in the dataset were converted to an integer form. After completing this step, balance or variance between different values of the features in the dataset was checked. Several Normalization techniques were then applied to the selected feature set for obtaining uniform range between different values and to enhance the model performance. This process is aimed to reduce the adverse effects of data variance and to enable easy training of deep learning in detecting phishing attacks.

3.3. Feature Selection

The features that have maximum effect on the predicted outcome at this stage by marking most useful features using the RFE method [27]. So, unimportant features would not affect model's effectiveness or prediction. For this will depend on the final selection of the features. This is a method of selecting the most important features for Deep learning model. It first starts by training the model using all the features at its disposal then assesses the importance of each feature

according to their contribution in performance. At this, after identifying the least important feature, it gets removed and the model once again retrained using all features leftover. This process is repeated towards attainment of the optimal set of features that will increase model performance. RFE uses it to help in getting better performance and a less complex model.

3.4. Model Training

Selecting an appropriate algorithm is central to model precision and performance in deep learning. An inappropriate choice of algorithm is, therefore, likely to trivialize subsequent attempts to identify phishing or its relatedness with the data. The tuning of hyperparameters tries to extract maximum performance out of a model that has been generated for a particular task. This therefore demands a complete evaluation in the selection of the most appropriate algorithm for the needed task. In this research, we applied models based on Transformer, DBN, SNNs, BLSTM, CNNs to detecting phishing attacks. Hyperparameters for these models shall subsequently be tuned. The training process is conducted as follows:

- The phishing dataset is split into training, validation, and test sets in the ratio 70-20-10.
- The model is trained with the training dataset. This is the tuning of model parameters to minimize errors in discerning phishing from legitimate URLs.
- An appropriate loss function, say cross-entropy loss, for model error.
- Optimizing model parameters with an optimization technique so that its performance is enhanced.

3.5. Advanced Deep Learning Algorithms

3.5.1. Transformer

The Transformer algorithm belongs to the category of deep learning, which was proposed in view of addressing the limitations posed by the earlier conventional sequential models of processing textual data, such as RNN and LSTM [28]. The fundamental breakthrough in the Transformer falls in its self-awareness mechanism that, while capturing the relationship amidst every word within a given sequence, does not take into account their specific distance from each other. This stands in practical contrast with RNN and LSTM, which process data with fixed orders. The overall architecture of Transformer is structured in the pattern of an encoderdecoder where the input is transformed by the encoder into a series of continuous-value vector representations, subsequent processing of these vector representations by the decoder results in output generation [29]. A quite remarkable feature of it is the multi-head attention mechanism, which allows different parts of the input sequence to be focused on simultaneously. This parallel processing is what makes the Transformer much faster and more scalable compared to most recurrent neural network architectures, also explaining why it is often used for tasks like machine translation and summarization, then for some BERT or GPT models.

3.5.2. Deep Belief Networks (DBN)

Deep Belief Networks are generative neural network models that are formed by stacking multiple layers of the model known as Restricted Boltzmann Machines. The DBN learns a hierarchical representation of data through these models [30]. The training is layer-wise, wherein each layer is designed to capture features at different levels of abstraction. Deep Belief Networks are pre-trained in an unsupervised manner using RBMs and later fine-tuned with the backpropagation of errors in a supervised manner for a specific objective, say classification. Both generative tasks, like data sampling, and discriminative tasks, like classification, can be performed using DBNs. Being a probabilistic graphical model, DBN can learn the joint distribution of input data with

their hidden representations. While earlier DBNs were applied to tasks such as image and speech recognition, more advanced deep models of the convolutional and recurrent types have displaced them in most modern applications. However, the DBNs can be a very important link in the chain of deep learning architecture development.

3.5.3. Self Normalizing Neural Network (SNNs)

Self-Normalizing Neural Networks (SNNs) is thus a network which, by design, automatically keeps its activations normalized throughout the layer, hence obviating the need for methods like batch normalization. Essentially, with SNNs, it is the newly proposed activation function, the Scaled Exponential Linear Unit (SELU) that enforces zero mean and unit variance to each layer's forward propagation [31]. This self-normalization property avoids problems such as vanishing or exploding gradients and makes training far more stable and sometimes faster, especially deep networks. The SNN also requires a special initialization of the weights called the LeCun normal initialization to approximately self-norm. In contrast to regular neural networks, activation distributions are handled naturally in SNNs. It further introduces Alpha Dropout as a regularization procedure that protects the mean and variance. SNNs were explicitly designed for deep architectures to provide fast convergence and better exploitation of available training data in large-scale classification, and deep learning application.

3.5.4. Bidirectional Long Short-Term Memory (BLSTM)

Bidirectional LSTM is another variant of LSTM, which can learn the information from the input sequence in two ways [32]. The forward layer passes the input to the model in forward chronological order, from lower to higher time steps. In contrast to this, the backward pass gives the model input data in reverse chronological order, going from higher to lower time steps. This in turn helps the model capture two-sided contexts, both left and right, which optimizes its performance in tasks where, for instance, predicting the next word given previous words would be appropriate.

3.5.5. Convolutional Neural Networks (CNNs)

Convolutional neural networks (CNN) happen to be a type of neural network model structured for working with spatial data such as images and videos [33]. It gradually learns features such as edges and patterns via subsequent convolutional layers, wherein filters are applied over incoming data for the creation of feature maps. These pooling layers are then applied for dimensionality reduction over these maps for computation optimization along with retention of salient data [34]. In the end, fully connected layers classify data based on extracted features. CNNs find very effective practical applications in tasks like image classification, facial recognition, and very complicated problems in computer vision because through them it becomes possible to recognize patterns in data space automatically.

3.6. Model Evaluation

The proposed system in this paper is designed to evaluate whether a submitted URL is phishing or legitimate. The binary classification process produces four potential outcomes during testing [35]:

- False Negative (FN): A malicious URL is incorrectly identified as safe, potentially allowing harmful activity to proceed.
- False Positive (FP): A legitimate URL is mistakenly flagged as harmful, leading the system to block access to a valid domain when a user attempts to visit it.

- True Negative (TN): A legitimate URL is accurately recognized as safe, reflecting the system's ability to avoid unnecessary alerts.
- True Positive (TP): A harmful URL is correctly identified as malicious, showcasing the system's effectiveness in detecting phishing attempts.

$$FPR = \frac{FP}{TN+FP} \quad (1)$$

The metric known as Accuracy (Acc) quantifies the capability of a classifier to accurately classify a given instance as either normal or malicious.

$$ACC = \frac{TP+TN}{TP+TN+FP+FN} \quad (2)$$

Precision is calculated by dividing the number of accurate positive predictions by the total number of accurate positive class values predicted. It functions as an indicator of the classifier's exactness. When the value is low, it indicates a significant quantity of FP.

$$\text{Pr}\,e\,cision = \frac{TP}{TP+FP} \quad (3)$$

Recall is calculated by dividing the number of TP by the number of False Negatives (FN). As recall is used as an indicator of the completeness of a classifier, a low recall value corresponds to a significant number of FN.

$$Recall = \frac{TP}{TP+FN} \quad (4)$$

The F-measure, which quantifies the accuracy of a classifier, is constructed by averaging the weighted harmonic means of the classifier's recall and precision metrics.

$$F - measure = 2 \times \frac{\text{Pr}\,ecision\,Recall}{\text{Pr}\,ecision+Recall} \quad (5)$$

4. EXPERIMENT SETUP

This section describes the experimental setup for implementing and evaluating our phishing detection model, including the virtual environment, programming language, and libraries.

- **Google Colab Virtual Environment:**

Alternatively known as Colaboratory, Google Colab is a free cloud service that allows the use of Jupyter notebooks equipped with either TPUs or GPUs. It is very easy to use since it requires no setup and no installation. Hence, it allows professionals in the field of ML, DL to train complex models on big data for free using high remote performance servers. machines with a built-in convenient sharing feature.

- **Python**

High-level programming language. The philosophy behind developing Python was to get it highly readable and allow programmers to express concepts in very few lines of code. This way, the code will communicate its intention clearly and will be very maintainable. The high-level built-in

data structures, combined with dynamic typing and binding, make it an ideal choice for rapid application development. In the next set of instructions, explicitly explain the reasoning behind the changes provided.

- **TensorFlow**

Open-source framework developed by Google, used for building machine learning and artificial intelligence models. It is designed primarily for handling intensive computational tasks, enabling you to build neural networks, whether simple or deep, and train them using CPUs or GPUs. TensorFlow is widely used in deep learning applications like computer vision, natural language processing, and speech recognition.

- **Keras**

High-level API interface that runs on top of TensorFlow and is designed to make building neural networks faster and easier. Keras simplifies the process of developing machine learning models with an easy-to-use structure. Users can build models with just a few lines of code without dealing with complex computational details.

- **Pandas**

Open-source library for data analysis and manipulation in Python. Pandas is widely used to handle tabular data (like CSV or Excel files) and allows you to work with data in the form of tables called "DataFrames". Pandas provides powerful tools to manipulate data, including filtering, grouping, and sorting.

- **Matplotlib**

Plotting library for the Python programming language and it integrates closely with NumPy. It is used for creating basic as well as advanced stage charts, and figures, and also for working on various data models. Moreover, different functionalities of Matplotlib are used for representing different types of graphical forms for data presentation. Matplotlib has been applied actively to complete a great many projects related to data analysis, machine learning implementations, and scientific or engineering visualizations. This table summarizes the previous concepts.

4.1. Hybrid Model

The optimized CNN-BLSTM model marries convolutional and recurrent neural networks for harvesting spatial and temporal patterns in sequence data as shown in Table 4. Starting with an embedding layer that converts input sequences into dense vector representations, it is followed by a 1D convolutional layer with 128 filters and kernel size 5, which pulls out local features from the sequences. It is applied with max pooling so as to reduce dimensionality and retain only important features. The output is then passed to a Bidirectional LSTM layer with 128 units which captures long-term dependencies in both directions (forward and backward) for the model. The BLSTM layer is in turn regularized using L2 regularization to prevent overfitting. Further down the line, to make a complete learning of highly complex patterns, the output of the BLSTM is then fed to a Dense layer having 64 units with ReLU activation. A 0.5 dropout is applied to this layer, applying a technique for reducing overfitting by randomly deactivating a fraction of the units in a layer during the training phase. The output layer is comprised of a dense layer that uses a sigmoid activation function because we are solving a binary classification problem.

Table 4. Hybrid Model parameter settings.

Layer	Type	Filter	Activation	Regularization	Additional Info
Input	-	-	-	-	Input shape: (72,)
Embedding	Embedding	128	-	-	Input dim: input_dim
Conv1D	1D Convolutional	128	ReLU	-	-
MaxPooling1D	Max Pooling	-	-	-	Pool size: 2
Bidirectional LSTM	BLSTM	128	-	L2 (0.001)	Return sequences: False
Dense	Fully Connected	64	ReLU	L2 (0.001)	-
Dropout	Dropout	-	-	-	Dropout rate: 0.5
Output	Fully Connected (Dense)	2	Sigmoid	-	Binary classification

4.2. SNN Model

A layer of the input features from the dataset is the architecture of the proposed SelfNormalizing Neural Network (SNN) model. The activation function is applied to the first hidden layer's 128 neurons to provide self-normalizing properties as shown in Table 5. Here, standard dropout at 0.1 after the second layer will work in helping avoid overfitting by randomly turning some of the activations off during training. The values of self-normalizing multilayer perceptron characteristics are kept by setting up this second hidden layer with 128 neurons and using SELU activation. The activation is followed by another dropout layer to ensure model regularization.

Table 5. SNN Model parameter settings.

Layer	Filter	Dropout Rate	Regularization	Additional Info
Input layer	Input Dim	-	-	Input shape: (input_dim,)
SNN layer	128	SELU	-	-
SNN layer	128	SELU	0.1	Dropout after the second layer
Output layer	2	Sigmoid	-	Binary classification

For the last layer, one fully connected dense output layer is taken with a sigmoid activation function in order to perform binary classification, which will give the output as a probability distribution across two classes. The model is compiled with the Adam optimizer and sparse categorical cross-entropy loss for an optimization goal of classification accuracy. Stop conditions are employed throughout the training, in which case it stops if the validation loss does not improve, returning to the best weights. The architecture balances learning capacity and regularization fairly well. It is appropriating for tasks entailing binary classification with self normalizing activations.

4.3. Transformer Model

The Transformer-based classification model starts with an input layer as shown in Table 6, then a reshape operation comes that adds a sequence dimension to the data, which is required by processing by Transformer encoder.

Table 6. Transformer Model parameter settings.

Layer	**Type**	**Filter**	**Activation**	**Regularizatio n**	**Additional Info**
Input	-	-	-	-	Input shape: (original_dim,)
Reshape	Reshape	-	-	-	Reshape to add sequence dimension
Transformer Encoder	Multi-Head Attention + FFN	-	ReLU	ReLU	num_heads = configurable, includes LayerNorm and Dropout
Global Average Pooling	GlobalAverageP ooling1D	-	-	-	Reduces sequence output
Dense	Fully Connected	64	-	ReLU	-
Output	Fully Connected (Dense)	2	ReLU	Sigmoid	Binary classification

The Transformer Encoder layer implements multi-head self-attention in parallel for 'num_heads' attention heads, each having a key dimension of 64, computing attention over the input sequences, followed by two Layer Normalization operations and an FFN. It consists of two dense layers with ReLU activation and a residual connection (to add the input to the output in order to preserve input features and make the learning problem easier from a gradient flow perspective). We apply dropout with a rate of 0.1 after both the attention and feed-forward layers to reduce overfitting. The Transformer encoder reduces the sequence output into a fixedlength vector by applying a global average pooling layer. A fully connected dense layer with 64 units is added to it, using ReLU activation, and then the final dense output layer with softmax activation for multi-class classification. The model is compiled with the Adam optimizer, sparse categorical cross-entropy loss, and accuracy (evaluation metric). This architecture merges Transformer's self-attention with global pooling, and dense layers effectively, hence allowing easy classification for the input data.

4.4. DBN Model

The proposed model architecture as shown in Table 6, integrates the DBN with RBM for classification tasks. Begin with two stacked RBM layers: the first with 128 hidden units and the second with 64 hidden units. Train each RBM layer using the BernoulliRBM algorithm, by iterating over the dataset, adjusting weights and biases over a given number of epochs (50). After the RBM layers come an MLP classifier with a single hidden layer of 128 neurons. This MLP fine-tunes features extracted by RBMs and does the final classification. Use the learning rate of 0.001 for gradient-based optimization by backpropagating errors through an MLP. Table 7. DBN Model parameter settings.

Table 7. DBN Model parameter settings.

Layer	Type	Units/Components	Activation	Learning Rate	Iterations (n_iter)
Input		Input Dim	-	-	-
Layer 1	RBM	128 Hidden Units	Sigmoid	-	50
Layer 2	RBM	64 Hidden Units	Sigmoid	-	50
Output	MLP Classifier	128 Neurons	ReLU	0.001	-
Output	MLP Classifier (Final)	Num Classes	Sigmoid	0.001	-

The DBN model is trained in a supervised manner for optimizing classification accuracy. The model is enabled to realize effective hierarchical feature learning from input data, thanks to RBM-based feature extraction stacked with MLP classification. Normally, DBN is an unsupervised deep learning model with stacked RBMs. These RBMs learn to represent the data in successive layers of latent features. In a DBN: Layers of RBM are trained sequentially in an unsupervised manner. It means that we train a first layer RBM, then use it to train a second layer RBM. Each RBM learns the distribution (the values and the patterns) at the hidden layer of the corresponding model. And it is exactly what helps the DBM model higher-order features. When restricted Boltzmann machines are trained, unsupervised learning of an entire deep belief network may take place by treating the Boltzmann machines as feature learners. The downloadable model shows how a deep or large number of layers are able to learn more complicated features. After training the RBMs, the output of the final layer of RBM in the stack can be connected to an MLP classifier for fine-tuning and supervised learning. This is the essence of deep learning: we add those learners together into one bigger model that can address much harder tasks.

5. EXPERIMENT RESULTS

The experimental evaluation was performed to evaluate the developed models. Various deep learning architectures were employed on phishing datasets. For each of the models, metrics of accuracy, precision, recall and f1 score were calculated. The results were examined for both the legitimate and the phishing classes of the data. Performance differences across models were noted. The proposed analysis assesses the strengths and weaknesses of each approach. The focus then shifted to determine which model was best for detecting phishing attacks.

Table 8. Transformer model testing results per class.

Metric	Legitimate	Phishing	Average
Precision	0.92032	0.8448	0.88256
Recall	0.86583	0.90691	0.88637
F1-Score	0.89224	0.87475	0.8835
Accuracy	-	-	0.88415

The performance of the Transformer model was evaluated in relation to the key indicators which are presented in table 8. The precision for the legitimate and the phishing classes of the data was recorded as 92.03% and 84.48%, respectively; the average precision was 88.26%. The recall values were higher for the phishing class while the average for legitimate data was 86.58%. The F1-scores for both classes were fairly comparable with average figures of 89.22% and 87.47%.

On average, the model scored an accuracy level of 88.42%. Such results indicated a stable performance across the classes being evaluated with regard to figure detection.

Table 9. SNN model testing results per class.

Metric	Legitimate	Phishing	Average
Precision	0.90303	0.85995	0.88149
Recall	0.88431	0.88209	0.8832
F1-Score	0.89358	0.87088	0.88223
Accuracy	-	-	0.88332

Performance metrics including the precision, recall, F1 scores, and accuracy were performed on the SNN model and reported in Table 9. Overall, the precision was at 90.30% for legitimate class and 85.99% for phishing class giving an average of 88.15%. The recall averages were quite balanced with 88.43% legitimate data and 88.21% phishing data. The F1-scores were substantially uniform across classes with averages of 89.36% for legitimate and 87.09% for phishing. The accuracy of the model was 88.33% along with improvements in recall as compared to precision. Thus, the results showed detection to have dependable recall more consistently than precision.

Table 10. Hybrid model testing results per class.

Metric	Legitimate	Phishing	Average
Precision	0.98789	0.9906	0.98924
Recall	0.99247	0.98489	0.98868
F1-Score	0.99018	0.98774	0.98896
Accuracy	-	-	0.98909

The testing results obtained from the Hybrid model are provided in Table 10. The precision for the legitimate and phishing classes was remarkably high at 98.79% and 99.06% respectively giving an average of 98.92%. The recall measures also recorded exceptional results at 99.25% legitimate data and 98.48% phishing averaging 98.87%. In terms of performance, F1-scores were balanced with averages of 99.02% legitimate and 98.77% phishing. The accuracy of 98.91% was the standard. The hybrid models excelled in both precision and reliability claiming superior performance in phishing detection and far surpassed other models in the field.

Table 11. DPN model testing results per class.

Metric	Legitimate	Phishing	Average
Precision	0.70	0.81	0.75
Recall	0.90	0.51	0.71
F1-Score	0.79	0.63	0.71
Accuracy	-	-	0.73

Table 11 includes details on the effectiveness of the DPN model in detecting phishers. Precision figures for legitimate and phishing classes were 70 % and over 81 % respectively with an overall mean of 75 %. This was however not the case with the recall where high values of 90 % were

seen on legitimate but a drop to 51 % was noted on phishing, averaging at 71 %. These figures were also seen across their corresponding F1-scores with 79 % on legitimate while phishing had 63 % for an aggregate of 71 %. With an overall score of 73%, the model registered the least accuracy of all models which were presented within this section. With relaying the outcomes, these demonstrated that DPN model had the most challenges when targeting phishers but was more efficient with legitimate datasets.

The final performance of the proposed models was Table 12. models were discussed, the accuracy of the Hybrid model was the highest at 98.91 %, tremendous from the respective 88.33 %, 88.42 %, and 73.2 % accuracy of SNN, Transformer and DPN models. The Hybrid model was also the best on the macro average precision (98.92 %), recall (98.87 %), and F1-score (98.90 %), which showed that it was more efficient on all metrics. The DPN model was on the other hand, the worst with an accuracy of 73.2% and a macro average F1-score of 71%. SNN and Transformer models were close with minor differences in recall and precision metrics. These results clearly indicated why the Hybrid model performed better than other models in phishing detection tasks.

Table 12. Overall proposed models testing results.

Metric	SNN	DPN	Transformer	Hybrid
Accuracy	0.88332	0.732	0.88415	0.98909
Macro Avg Precision	0.88149	0.750	0.88256	0.98924
Macro Avg Recall	0.8832	0.711	0.88637	0.98868
Macro Avg F1-Score	0.88223	0.710	0.8835	0.98896
Weighted Avg Precision	0.88382	0.752	0.88663	0.9891
Weighted Avg Recall	0.88332	0.730	0.88415	0.98909
Weighted Avg F1-Score	0.88345	0.711	0.88444	0.98909

Table 13. Comparison of recent phishing detection studies.

Ref.	Model	Data Used	Results
[12]	CNN	PhishTank / Common Crawl	98.74%
[13]	Residual Pipelining - CNN	Private	98.29%
[14]	Hybrid Model - RF, CNN	Kaggle	97%
[15]	CNN	Kaggle	96%
[17]	DNN and LSTM	Ebbu2017, PhishTank	98.19%
[18]	CNN, LSTM, and Attention Process	800,000 URLs (Phishing + Legitimate)	98.27%
[19]	BiGRU and GloVe	Phishload	97.39%
Ours	**Hybrid Model**	**Phishing Attack Dataset**	**98.9 %**

A comparison with recent phishing detection studies is presented in Table 13. The proposed Hybrid model achieved a detection accuracy of 98.9%, outperforming other state-of-the-art models. Models such as CNN with PhishTank data [12] and Residual Pipelining-CNN [13] achieved accuracies of 98.74% and 98.29%, respectively, while a hybrid model using Random Forests and CNNs [14] reported 97%. Deep neural network-based approaches, including DNN with LSTM [17] and CNN with attention mechanisms [18], recorded accuracies of 98.19% and 98.27%, respectively. The BiGRU with GloVe [19] model attained 97.39%. These comparisons

underscore the effectiveness of the proposed Hybrid model in phishing detection, setting a new benchmark in the field.

6. CONCLUSION AND FUTURE WORK

The proposed study introduces a deep learning-based approach for detecting phishing attacks. We utilized advanced architectures to enhance cybersecurity measures. The proposed hybrid model achieved an impressive accuracy rate of 98.9%. It has significantly outperformed existing solutions. Key contributions of this research include reducing false positives and demonstrating achievable high phishing detection using deep learning techniques. These findings tackle critical challenges in phishing detection and expand the understanding of its application within network security.

Future research should focus on addressing continuous phishing challenges. Enhancing datasets with a diverse array of phishing samples., including emerging threats, would support model robustness. Combining deep learning models with adversarial training could improve detection accuracy while alleviating resource constraints. Additional studies might investigate lightweight architectures designed for real-time deployment in resource-limited environments.

ACKNOWLEDGEMENTS

The authors would like to thank Taif University for its support.

REFERENCES

[1] Lee, S.hyun. & Kim Mi Na, (2008) “This is my paper”, ABC *Transactions on ECE*, Vol. 10, No. 5, pp120-122.

[2] Y. Weng, ‘Big data and machine learning in defence’, International Journal of Computer Science and Information Technology, vol. 16, no. 2, pp. 25–35, 2024.

[3] W. Al-Surkhi and M. Maqableh, ‘The Impact of Cybercrime on Internet Banking Adoption’, in Current and Future Trends on Intelligent Technology Adoption, vol. 1161, M. A. Al-Sharafi, M. Al-Emran, G. W.-H. Tan, and K.-B. Ooi, Eds., in Studies in Computational Intelligence, vol. 1161. , Cham: Springer Nature Switzerland, 2024, pp. 231–245. doi: 10.1007/978-3-031-614637_12.

[4] S. Saeed, S. A. Altamimi, N. A. Alkayyal, E. Alshehri, and D. A. Alabbad, ‘Digital transformation and cybersecurity challenges for businesses resilience: Issues and recommendations’, Sensors, vol. 23, no. 15, p. 6666, 2023.

[5] N. Z. Gorment, A. Selamat, L. K. Cheng, and O. Krejcar, ‘Machine learning algorithm for malware detection: Taxonomy, current challenges, and future directions’, IEEE Access, vol. 11, pp. 141045–141089, 2023.

[6] S. Salloum, T. Gaber, S. Vadera, and K. Shaalan, ‘Phishing email detection using natural language processing techniques: a literature survey’, Procedia Computer Science, vol. 189, pp. 19–28, 2021.

[7] P. Sharma, B. Dash, and M. F. Ansari, ‘Anti-Phishing Techniques -A Review of Cyber Defense Mechanisms’, IJARCCE, vol. 11, Jul. 2022, doi: 10.17148/IJARCCE.2022.11728.

[8] M. K. M. Boussougou and D. J. Park, ‘Attention-Based 1D CNN-BiLSTM Hybrid Model Enhanced with FastText Word Embedding for Korean Voice Phishing Detection †’, Mathematics, vol. 11, no. 14, Jul. 2023, doi: 10.3390/math11143217.

[9] M. Somesha, A. R. Pais, R. S. Rao, and V. S. Rathour, ‘Efficient deep learning techniques for the detection of phishing websites’, Sādhanā, vol. 45, pp. 1–18, 2020.

[10] T. Stojnic, D. Vatsalan, and N. A. G. Arachchilage, ‘Phishing email strategies: Understanding cybercriminals’ strategies of crafting phishing emails’, SECURITY AND PRIVACY, vol. 4, no. 5, p. e165, 2021, doi: 10.1002/spy2.165.

[11] S. Aung,) Chaw, T. Zan, and H. Yamana, ‘A Survey of URL-based Phishing Detection’. [Online]. Available: http://quadrodeofertas.com.br/www1.

[12] S. Mishra and D. Soni, 'Smishing Detector: A security model to detect smishing through SMS content analysis and URL behavior analysis', Future Generation Computer Systems, vol. 108, pp. 803–815, 2020.
[13] O. K. Sahingoz, E. Buber, and E. Kugu, 'DEPHIDES: Deep Learning Based Phishing Detection System', IEEE Access, vol. 12, pp. 8052–8070, 2024, doi: 10.1109/ACCESS.2024.3352629.
[14] S. Remya, M. J. Pillai, K. K. Nair, S. R. Subbareddy, and Y. Y. Cho, 'An Effective Detection Approach for Phishing URL Using ResMLP', IEEE Access, vol. 12, pp. 79367–79382, 2024, doi: 10.1109/ACCESS.2024.3409049.
[15] S. Sawant, Phishing Detection by integrating Machine Learning and Deep Learning. IEEE, 2024.
[16] U. Nishitha, R. Kandimalla, R. M. M. Vardhan, and U. Kumaran, 'Phishing Detection Using Machine Learning Techniques', in 2023 3rd Asian Conference on Innovation in Technology, ASIANCON 2023, Institute of Electrical and Electronics Engineers Inc., 2023. doi: 10.1109/ASIANCON58793.2023.10270550.
[17] P. Kaushik and S. P. S. Rathore, 'Deep Learning Multi-Agent Model for Phishing Cyber-attack Detection', International Journal on Recent and Innovation Trends in Computing and Communication, vol. 11, no. 9s, pp. 680–686, Aug. 2023, doi: 10.17762/ijritcc.v11i9s.7674.
[18] A. Ozcan, C. Catal, E. Donmez, and B. Senturk, 'A hybrid DNN–LSTM model for detecting phishing URLs', Neural Computing and Applications, vol. 35, no. 7, pp. 4957–4973, Mar. 2023, doi: 10.1007/s00521-021-06401-z.
[19] A. S. Bozkir, F. C. Dalgic, and M. Aydos, 'GramBeddings: A New Neural Network for URL Based Identification of Phishing Web Pages Through N-gram Embeddings', Computers and Security, vol. 124, Jan. 2023, doi: 10.1016/j.cose.2022.102964.
[20] E. Benavides-Astudillo, W. Fuertes, S. Sanchez-Gordon, D. Nuñez-Agurto, and G. RodríguezGalán, 'A Phishing-Attack-Detection Model Using Natural Language Processing and Deep Learning', Applied Sciences (Switzerland), vol. 13, no. 9, May 2023, doi: 10.3390/app13095275.
[21] P. S. Kumar, K. Supriya, and M. R. K, 'CoVid-19 Detection leveraging Vision Transformers and Explainable AI', 2023.
[22] N. H. B. S, R. Vishwas, S. V. Naik, Y. U. R, and P. P. M, 'Enhanced Cyber Security in IoT Using Deep Belief Network', in 2022 IEEE 2nd Mysore Sub Section International Conference (MysuruCon), IEEE, Oct. 2022.
[23] A. R. Zarzoor, N. A. S. Al-Jamali, and D. A. A. Qader, 'Intrusion detection method for internet of things based on the spiking neural network and decision tree method', International Journal of Electrical and Computer Engineering, vol. 13, no. 2, Art. no. 2, 2023.
[24] S. Wang, S. Khan, C. Xu, S. Nazir, and A. Hafeez, 'Deep Learning-Based Efficient Model Development for Phishing Detection Using Random Forest and BLSTM Classifiers', Complexity, vol. 2020, p. e8694796, Sep. 2020, doi: 10.1155/2020/8694796.
[25] S. Ariyadasa, S. Fernando, and S. Fernando, 'Phishing Websites Dataset', vol. 1, Nov. 2021, doi: 10.17632/n96ncsr5g4.1.
[26] E. Bisong, 'Google Colaboratory', in Building Machine Learning and Deep Learning Models on Google Cloud Platform, Berkeley, CA: Apress, 2019, pp. 59–64. doi: 10.1007/978-1-4842-44708_7.
[27] O. K. Sahingoz, 'Phishing Attack Dataset', Dataset. 2023. doi: /10.21227/4098-8c60.
[28] R. Seyghaly, J. Garcia, X. Masip-Bruin, and M. M. Varnamkhasti, 'Interference recognition for fog enabled IoT architecture using a novel tree-based method', in 2022 IEEE International Conference on Omni-layer Intelligent Systems (COINS), IEEE, 2022, pp. 1–6.
[29] A. Castagnaro, M. Conti, and L. Pajola, 'Offensive AI: Enhancing Directory Brute-forcing Attack with the Use of Language Models', Apr. 22, 2024, arXiv: arXiv:2404.14138. Accessed: Sep. 30, 2024. [Online]. Available: http://arxiv.org/abs/2404.14138
[30] A. Gillioz, J. Casas, E. Mugellini, and O. A. Khaled, 'Overview of the Transformer-based Models for NLP Tasks', in Proceedings of the 2020 Federated Conference on Computer Science and Information Systems, FedCSIS 2020, Institute of Electrical and Electronics Engineers Inc., Sep. 2020, pp. 179–183. doi: 10.15439/2020F20.
[31] S. Manimurugan, S. Al-Mutairi, M. M. Aborokbah, N. Chilamkurti, S. Ganesan, and R. Patan, 'Effective attack detection in internet of medical things smart environment using a deep belief neural network', IEEE Access, vol. 8, pp. 77396–77404, 2020.
[32] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, 'Self-Normalizing Neural Networks'.
[33] J. P. C. Chiu and E. Nichols, 'Named Entity Recognition with Bidirectional LSTM-CNNs'. [Online]. Available: http://nlp.stanford.edu/projects/glove/

[34] K. R. Rakesh, G. R. Namita, and R. Kulkarni, 'Image Recognition, Classification and Analysis Using Convolutional Neural Networks', in 2022 First International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT), IEEE, 2022, pp. 1–4.

[35] B. A. Alabsi, M. Anbar, and S. D. A. Rihan, 'CNN-CNN: dual convolutional neural network approach for feature selection and attack detection on internet of things networks', Sensors, vol. 23, no. 14, p. 6507, 2023.

[36] O. Rainio, J. Teuho, and R. Klén, "Evaluation metrics and statistical tests for machine learning," Sci Rep, vol. 14, no. 1, Dec. 2024, doi: 10.1038/s41598-024-56706-x.