Applied Mathematics on Science and Engineering

1(1) (2024): 21 –35

Research Article

Evaluation of Deep Learning Algorithms in Comparison for Phishing Email Identification

Mohamed Hassan[1]

[1]Al- Azhar University, Faculty of Science, Egypt

Abstract:

Using skewed sequential data, the study explores the effectiveness of numerous sequential models designed for binary classification tasks. The dataset under investigation consists of 5,595 testing samples and 13,055 training samples, a structure that presents significant difficulties because of uneven labelling. The researchers carefully go through pretreatment procedures, which include text data encoding and effective methods for handling missing information, in order to address this. The study employs and examines a wide range of algorithms, which reflects the heterogeneous sequential modelling environment. A variety of neural network architectures are included in the arsenal CNN, CNN-RNN, RCNN. The binary classification job at hand is used to thoroughly assess each architecture, revealing both its advantages and disadvantages. The study's evaluation approach, which presents a wide range of measures indicating consistently excellent performance overall, is its key component. Among these algorithms stand out as the best with an astounding 97% accuracy rate on a variety of evaluation metrics. This strong performance highlights their ability to handle sequential data with unbalanced labels and establishes a standard for further work in related fields. Beyond its empirical results, the study is important because it provides a well-designed assessment approach that may be used as a benchmark by practitioners facing similar problems. Through the clarification of important concepts related to model selection and performance evaluation, the study provides professionals and academics with crucial resources to efficiently traverse the complex terrain of sequential modelling.

Keywords*:* Sequential Models, Binary Classification, Imbalanced Data, Model Evaluation, Confusion Matrix

Corresponding:

hassanmohamed85@gmail.com

Article History:

Received: March 28, 2024
Accepted: June 6, 2024
Published: June 30, 2024

How to Cite:

Hasan (2024). “Comparative Analysis of Deep Learning Algorithms for Phishing Email Detection”. *Applied Mathematics on Science and Engineering. 1* (1): 21 – 35

INTRODUCTION

Cybersecurity, as defined in references [1,2], encompasses the measures taken to safeguard the internet- connected resources against various forms of cyberattacks [3,4]. The ever-evolving landscape of cybersecurity presents an ongoing challenge in identifying, analyzing, and mitigating potential risks. Cyberattacks constitute digital malevolent efforts to pilfer, sabotage, or breach personal or confidential data, whether belonging to individuals or organizations [5]. A specific example of such an attack is phishing, where deceptive websites are employed to extract sensitive information from unsuspecting victims, including login credentials and credit card details.

Phishing is a kind of cyberattack characterized by the use of fraudulent emails, texts, or websites with the aim of stealing private data, such as bank account information and login credentials [6]. The rising prevalence of phishing attacks in recent years underscores the urgent need for both organizations and individuals to develop effective strategies for detecting and mitigating these threats. As noted by [7], the term "phishing" first appeared in 1996 during a social engineering attack against AOL (America Online) accounts by online scammers. Interestingly, the term's origins can be traced back to one of the earliest forms of hacking, known as "Phone Phreaking," which led to the adoption of "ph" as a common replacement for "f" in hacking terminology [8].

Spoofing is another term closely associated with phishing, encompassing domain spoofing, website spoofing, and email spoofing. According to [9], spoofing occurs when a malicious actor forges an email to mimic a legitimate sender's address. Since the email appears to originate from a trustworthy source, recipients often open it. Figure 3 provides a visual representation of the common phishing attack process. In [10] Phishing attacks usually start when an email with a phishing link or attachment is sent by the sender, who is frequently the attacker.

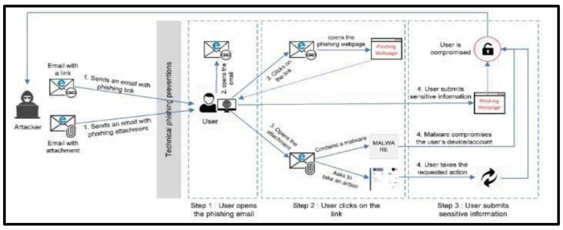
Email with a link
1. Sends an email with phishing link
Attacker
1. Sends an email with phishing attachment
Email with attachment
Technical phishing preventions
User
2. opens the email
3. Clicks on the link
opens the phishing webpage
Phishing Webpage
3. Opens the attachment
Contains a malware
MALWARE
Asks to take an action
User is compromised
4. User submits sensitive information
Phishing Webpage
4. Malware compromises the user's device/account
4. User takes the requested action
Step 1 : User opens the phishing email
Step 2 : User clicks on the link
Step 3 : User submits sensitive information

Figure 1. Illustrates the typical progression of a common phishing attack, as detailed by [10].

More than one study analyses phishing emails. Researchers in [11] present a comprehensive effort to enhance phishing detection through the utilization of advanced deep learning techniques. With existing methods facing challenges in performance accuracy and the identification of unknown attacks, the study delves into the domain of machine learning to tackle this issue. The researchers propose a taxonomy of deep learning algorithms for phishing detection, meticulously examining 81 selected papers using a systematic literature review approach. This study covers a wide range of deep learning models including DNN, MLP, CNN, RNN, RNN- RNN, LSTM, LSTM-LSTM, BiL STM-BiLSTM, GRU, GRU-GRU, BiGRU-BiGRU, and AE, with accuracy ranging from 91.27% to 96.83%. This exploration identifies their strengths and weaknesses, highlighting their potential in phishing detection. Despite the promising outcomes, challenges such as manual parameter tuning, prolonged training time, and suboptimal detection accuracy remain. This study not only offers valuable insights into the potential of deep learning for phishing detection but also pinpoints areas for future research to overcome these challenges.

In [12], researchers proposed a phishing email detection method leveraging deep semantic analysis and machine learning algorithms. They employed algorithms like NB, SVM, DT, LSTM, CNN, and Embedding on a dataset containing labeled emails. The study achieved accuracy rates ranging from 75.72% to 95.97%, highlighting the importance of in-depth analysis and machine learning in detecting phishing emails.

In [13], researchers tackled the problem of detecting phishing and spam emails using deep learning and natural language processing techniques. They utilized algorithms like LSTM and MLP on a dataset containing labeled messages. The study achieved accuracy rates of 99% for LSTM and 94% for MLP, showcasing the power of deep learning in enhancing email security.

Researchers in [14] addressed the challenge of data imbalance between phishing and benign emails. They proposed algorithms like DT, RF, GND, MLP, KNN, SEL, SVEL, FMPED, and FMMPED to achieve accurate detection. The study achieved impressive accuracy rates ranging from 90.25% to 99.45%, highlighting the importance of addressing data imbalance for effective detection.

In [15], researchers developed a solution for detecting email phishing attacks using algorithms like SVM, Naive Bayes, and LSTM. The study achieved high accuracy rates ranging from 97.00% to 99.62%, demonstrating the potential of machine learning in accurately identifying phishing attacks.

Researchers in [16] introduced algorithms to address the imbalance between phishing and benign emails. They employed techniques like DT, RF, GND, MLP, KNN, SEL, SVEL, FMPED, and FMMPED, achieving accuracy rates ranging from 88.50% to 99.45%. This study highlighted the importance of tackling data imbalance for effective phishing detection.

In [17], researchers developed a phishing email detection method using a real-world dataset and machine learning algorithms like NB, KNN, RF, SVM, and DT. The study evaluated the time taken for training and testing, achieving accuracy rates ranging from 96.669% to 99.451%. This research emphasized both accuracy and efficiency in phishing email detection.

Researchers in [18] propose a solution named RAIDER (Reinforcement AIded Spear Phishing DEtectoR) to address the challenges posed by spear phishing. These challenges include the difficulty of detection, susceptibility of machine learning to zero-day attacks, issues with email address spoofing, and scalability concerns. They conducted their study using a dataset comprising over 11,000 emails from three different attack scenarios. The proposed algorithm, RAIDER, is a reinforcement learning-based feature evaluation system that autonomously selects significant features to detect various spear phishing attacks. Notably, RAIDER achieves an enhancement in spoofing attack detection accuracy from 90% to 94%, and in Known Sender attack detection accuracy from 49% to 62%. Its strengths lie in the autonomous feature selection process and a remarkable 55% reduction in the required features' dimensions. However, its weaknesses include reliance on historical data access and potential limitations in identifying sophisticated attacks.

Related Works

In [19], authors super covenant analyzed the threats of phishing and spam emails with the help of deep learning and natural language processing tools. By training LSTM and MLP using a dataset with labeled messages they deployed it. This research reached an accuracy of 99% for LSTM and 94% for MLP, which demonstrates possibilities for improving repeated email security using deep learning.

Researchers in [14] further highlights the importance of the early detection in aging through the application of sophisticated technologies. Our proposed detection model is based on machine learning where the dataset was equally split into train and test sets. The features include email text, subject line, sender details, and URLs among others; the model either classifies emails as phishing or not. Analyzing the outcomes of the models built across three datasets indicated that the models with the largest feature vectors gave the highest accurate results, where model accuracy of the boosted decision tree model reached 0.88, 1.00, and 0.97, respectively. Also, the paper covers technical countermeasures and recommendations for the increase of users' awareness that remains critical for fighting against phishing.

In [15], the authors proposed a method of differentiating on email phishing attack from normal ones using algorithms such as SVM, Naive Bayes, and LSTM. The study recorded accuracies between 97% percent TO 99% and all of the chosen statistical techniques proved useful in analyzing the results of the study.

Researchers in [16] proposed algorithms to solve the problem of distinctive quantities of phishing and all normal messages. The analyzed data used methods such as DT, RF, MLP, KNN, SEL, SVEL, FMPED, and FMMPED. It has been reported to work with accuracies ranging from 88.50% to 99.45%. They expressed the need to address issues of data imbalance that this study has laid emphasis on in the fight against phishing.

Another work [20] that uses GCN in conjunction with NLP algorithms is the work that is specializing in the detection of phishing in the body texts of emails. The model has fairly tested satisfactory, yielding to an accuracy of 98% by using a self-generated email body text dataset. 2%. The novelty of the proposed approach is based on the fact that GCN is used for text classification for the first time and it is applied to an important and practical problem of phishing emails detection. Nevertheless, the main drawback of the study is that it was undertaken in an English setting, and the results might not be applicable to other languages or work- related emails.

In research [12] applies on collective dataset that used secure cloud-based cybersecurity solutions. This dataset contained 18,366 labelled emails, of which 3,416 were phishing emails and 14,950 were regular emails. CNN accuracy rate is 95.97 % and LSTMs networks are 95.91%. The novelty of this study is in using deep semantic analysis to capture inherent characteristics of the text body.

Work [21] helps to use the Multi-Layer Perceptron (MLP) model with the participation of Spam Base, Spam Assassin, and, finally, the UK.2011 Web spam datasets with the given accuracies 96.9%, 98.1%, and 95.6%, respectively. This work is merit worthy because of the several dataset and feature sets used in the research, thus the assessment of spam detection is more holistic. However, the main drawback of the study is the fact that it has focused squarely in the area of spam detection and not thoroughly on the special features of phishing, which may be a problematic area regarding the efficiency against phishing spam emails.

One other research work [22] that used CNN and LSTM networks compared the results between Adam and Stochastic Gradient Descent (SGD) in dataset contains 18,365 emails, of which 3,416 are phishing emails, wherein the study attained an accuracy of 98.3% for LSTM and 96.52% for CNN. The Adam optimizer was chosen as the optimizer to work with the given problem, which was shown to be more effective than SGD according to the results of the research mentioned earlier in this paper. But the comparison was done for only the classification of textual data, for a practical consideration, which can miss other important features like metadata and user behavior patterns in phishing detection.

METHODOLOGY

The dataset is divided into two sets: 13,055 samples with two attributes (X_train, y_train) and 5,595 samples (X_test, y_test) for testing. The binary labels in the 'Email Type' column are unbalanced within the dataset. In preprocessing, text data is tokenized, padded, and label-encoded in addition to replacing missing values in the 'Email Text' column. CNN, CNN-RNN, RCNN implemented algorithms that are specifically designed for binary classification problems. Model performance is evaluated using evaluation measures including Accuracy, Precision, Recall, and F1-Score in conjunction with a confusion matrix. The formulas that are supplied clarify how these measures are calculated. A thorough grasp of the dataset, preprocessing, algorithms, and assessment measures within the machine learning framework is provided by the presentation as a whole. Figure 1 shows model block diagram.

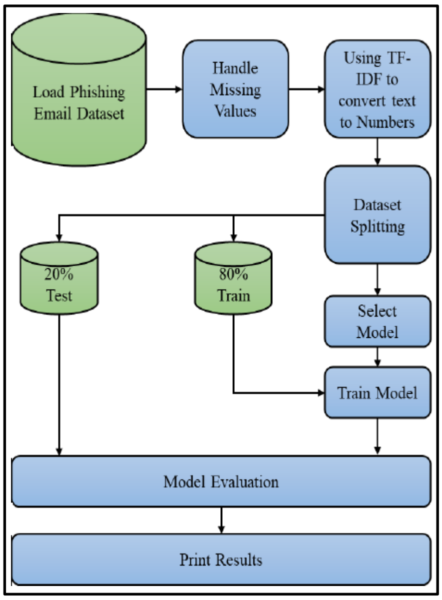

Figure 1. Model Block Diagram

Dataset Description

Two sets make up the dataset: 5,595 samples in the testing set (X_test, y_test) and 13,055 samples in the training set (X_train, y_train). Every sample in the training set has two features, and the 'y_train' array represents the binary labels for each feature. Likewise, the features of the testing set are kept in 'X_test,' and the binary labels that correspond to them are kept in 'y_test.' With 7,328 samples labelled as 1 in the training set and 11,322 samples labelled as 1 in the testing set, the dataset is unbalanced. The labels are of data type int64, as indicated by the 'Email Type' column. Binary classification is required for this task, most likely guessing the "Email Type" from the provided features. In order to guarantee strong performance on both classes, it is imperative to resolve the class imbalance during model training.

Dataset Preprocessing

During the preprocessing phase, NaN values in the 'Email Text' column of the DataFrame 'df' are substituted with an empty string ('') to address missing values. The text data is then ready for model training through the processes of tokenization and padding. Text sequences are converted into numerical representations using the Tokenizer fitted on the 'Email Text' column, where each distinct word is given an index. To guarantee uniform input dimensions for the model, the sequences are padded to a maximum sequence length of 100 words after the total number of unique words is determined. Applying label encoding with a LabelEncoder to the target variables 'y_train' and 'y_test' also converts categorical labels into numerical representations. You can use the encoded labels that are produced, 'y_train_encoded' and 'y_test_encoded,' to train and assess the machine learning model. Preparing text data and numerical labels for additional analysis and model development requires these preprocessing processes.

Implemented Algorithms

CNN

The sequential model shown uses a combination of denser layers, global max pooling, 1D convolutional layers, and embedding layers for binary classification. Beginning with an Embedding layer that, using a vocabulary of 16,660,400 words, converts input sequences of numbers into dense vectors of dimensionality 100.

Local patterns in the data are then captured by a Conv1D layer with 128 filters and a window size of 5. The most prominent features from the entire sequence are extracted using the Global Max Pooling1D layer.

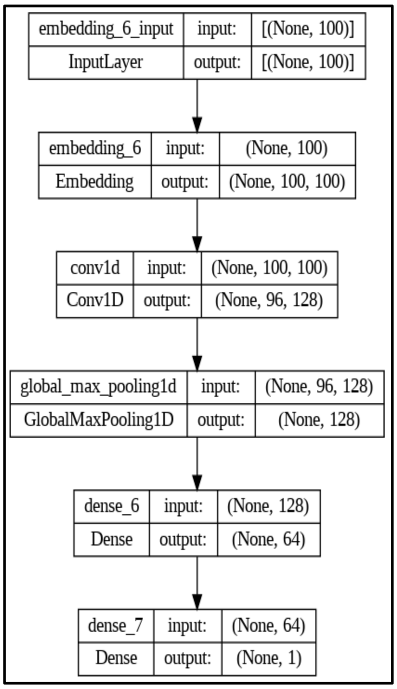

Figure 2. CNN Model

Repaired linear unit (ReLU) and sigmoid activation functions are then used by two Dense layers with 64 and 1 units, respectively, for additional processing and binary classification. There are 16,732,849 parameters in all for the model, and they can all be trained.This architecture is appropriate for binary classification tasks involving sequential data since it makes use of convolutional and pooling layers to capture hierarchical features within the input sequences.

CNN-RNN

The Sequential model is an intricate architecture designed for binary classification that consists of three Convolutional layers with increasing kernel sizes (5) and decreasing filter sizes (128, 64, and 32) that capture various local patterns. The Embedding layer comes first in the Sequential model. The sequential data is then processed by an LSTM layer that has 64 units and return sequences enabled. The most notable features are extracted from the full sequence using a Global Max Pooling layer. Two Dense layers follow, with the first employing ReLU activation and 64 units, and the second using a sigmoid activation for binary classification.

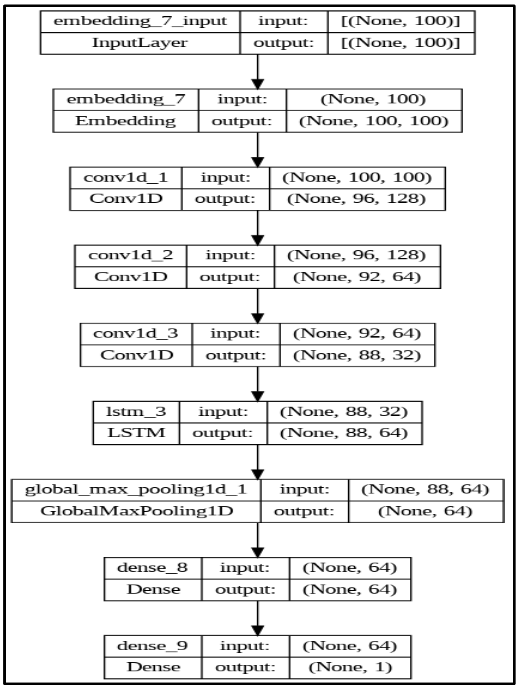
embedding_7_input | input: | [(None, 100)]
InputLayer | output: | [(None, 100)]
embedding_7 | input: | (None, 100)
Embedding | output: | (None, 100, 100)
conv1d_1 | input: | (None, 100, 100)
Conv1D | output: | (None, 96, 128)
conv1d_2 | input: | (None, 96, 128)
Conv1D | output: | (None, 92, 64)
conv1d_3 | input: | (None, 92, 64)
Conv1D | output: | (None, 88, 32)
lstm_3 | input: | (None, 88, 32)
LSTM | output: | (None, 88, 64)
global_max_pooling1d_1 | input: | (None, 88, 64)
GlobalMaxPooling1D | output: | (None, 64)
dense_8 | input: | (None, 64)
Dense | output: | (None, 64)
dense_9 | input: | (None, 64)
Dense | output: | (None, 1)

Figure 3. CNN-RNN Model

The model, with a total of 16,804,881 trainable parameters, adeptly combines Convolutional and Recurrent layers, offering a robust framework for capturing intricate relationships within sequential data and achieving high- performance binary classification.

RCNN

The presented model is a complex architecture designed for sequence processing and binary classification. It begins with an input layer of size 100, followed by an Embedding layer, mapping input sequences of integers into dense vectors of dimensionality 100. A Conv1D layer with 128 filters and a window size of 5 captures local patterns in SimpleRNN layer with 64 units processes the pooled data, preserving sequential information. The model concatenates the outputs of the MaxPooling1D and SimpleRNN layers along the sequence axis, creating a fused representation. The concatenated output is then flattened and connected to a Dense layer with a sigmoid activation function for binary classification. The model comprises a total of 16,741,489 parameters, all of which are trainable, indicating a substantial capacity for learning intricate patterns within sequential data.

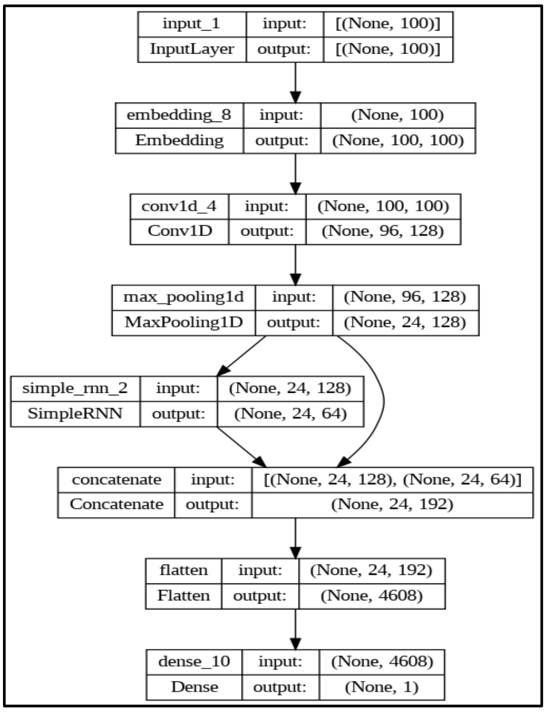
input_1 | input: | [(None, 100)]
InputLayer | output: | [(None, 100)]
embedding_8 | input: | (None, 100)
Embedding | output: | (None, 100, 100)
conv1d_4 | input: | (None, 100, 100)
Conv1D | output: | (None, 96, 128)
max_pooling1d | input: | (None, 96, 128)
MaxPooling1D | output: | (None, 24, 128)
simple_rnn_2 | input: | (None, 24, 128)
SimpleRNN | output: | (None, 24, 64)
concatenate | input: | [(None, 24, 128), (None, 24, 64)]
Concatenate | output: | (None, 24, 192)
flatten | input: | (None, 24, 192)
Flatten | output: | (None, 4608)
dense_10 | input: | (None, 4608)
Dense | output: | (None, 1)

Figure 4. RCNN Model

Evaluation Metrics

In the context of the article, the evaluation metrics— Accuracy, Precision, Recall, and F1-Score—serve as crucial measures to assess the performance of a machine learning or classification model. A confusion matrix accompanies them to understand the model's behavior [23] comprehensively.

Accuracy

Accuracy is a fundamental evaluation metric that measures the proportion of correctly classified instances out of the total instances. It provides an overall indication of the model's correctness in making predictions. Higher accuracy values indicate more effective model performance, but more is needed when dealing with imbalanced datasets.

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

Precision

focuses on the model's ability to make accurate, optimistic predictions. It is calculated as the ratio of true positives to the sum of true negatives and false positives. A high precision score implies that the model is less likely to make false positive errors, which can be crucial when such errors are costly.

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

Recall

Recall, also known as Sensitivity or True Positive Rate, measures the ability of the model to identify all positive instances correctly. It is computed as the ratio of true positives to the sum of true positives and false negatives. High recall is essential when it is crucial not to miss positive cases.

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

F1-Score

The F1-Score is a harmonic mean of precision and recall. It balances these two metrics, providing a score that considers false positives and false negatives. The F1-Score is particularly useful when you want a comprehensive evaluation that considers precision and recall simultaneously.

$$1 - Score = 2 \times \frac{percision \times recall}{Percison + recall}$$

Confusion Matrix

The confusion matrix visually represents a model's performance, especially in binary classification. It categorizes the model's predictions into four categories: true positives, true negatives, false positives, and false negatives. It offers an in-depth understanding of where the model excels and where it falters in its predictions.

RESULTS AND DISCUSSION

CNN Results

Important insights regarding the CNN model's classification performance can be gained by examining the confusion matrix that is presented in Figure 5. The model exhibits efficacy in accurately recognizing positive and negative cases, with 1395 true positives (TP) and 2188 true negatives (TN). False positives (FP) are occasions when the data is incorrectly classified as positive and false negatives (FN) are cases when the data is incorrectly classified as negative.

According to Confusion Matrix in Figure 5 the evaluation metrics for model testing calculated for each class of the data as shown in Table 1.

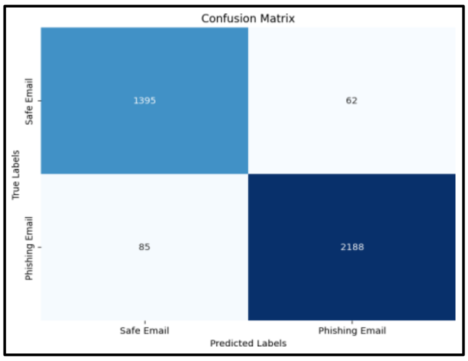

Figure 5. CNN Model Confusion Matrix

Table 1. CNN Classification Report

Class	Accuracy	Precision	Recall	F-Score
Safe Mail	0.97	0.96	0.96	0.96
Phishing Mail	0.97	0.98	0.97	0.97
Macro Avg	0.97	0.97	0.97	0.97
Weighted Avg	0.97	0.98	0.97	0.97

CNN- RNN Results

With 1422 true positive (TP) predictions and 2155 true negative (TN) predictions, the model performs well in accurately recognizing instances of both positive and negative classes, according to an analysis of the confusion matrix in Figure 6. On the other hand, 35 false positive (FP) predictions. show that negative samples were mistakenly identified as positive. Furthermore, 188 false negative (FN) predictions in the model indicate cases in which positive samples were incorrectly categorized as negative. The model performs admirably overall, with a balanced capacity to categories both positive and negative cases, despite a somewhat large proportion of false negatives.

According to Confusion Matrix in Figure 6 the evaluation metrics for model testing calculated for each class of the data as shown in Table 2.

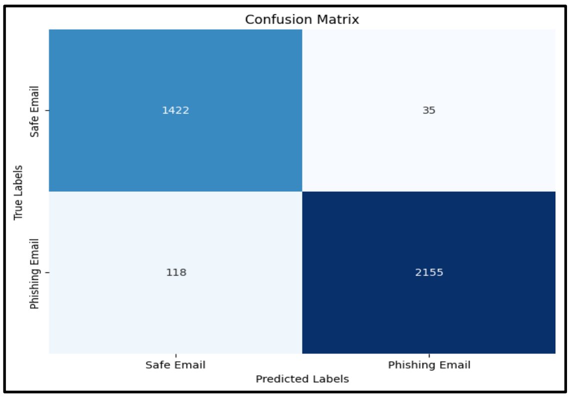

Figure 6. CNN-RNN Model Confusion Matrix

Table 2. CNN-RNN Classification Report.

Class	Accuracy	Precision	Recall	F-Score
Safe Mail	0.96	0.92	0.98	0.95
Phishing Mail	0.96	0.98	0.95	0.97
Macro Avg	0.96	0.95	0.96	0.96
Weighted Avg	0.96	0.96	0.96	0.96

RCNN Results

Its ability to correctly categories positive examples is demonstrated by the Confusion Matrix of the RCNN model in Figure 7, which correctly classifies 1475 instances of the positive class (TP). Furthermore, it correctly classifies 2123 cases as negative (TN), indicating that it can recognize negative instances with accuracy.

On the other hand, 43 false positive (FP) predictions are also made by the model, which mistakenly classifies negative cases as positive. This implies some misclassification mistakes, even though there are very few false positives in relation to the entire sample size.

In addition, the model produces 86 false negative (FN) predictions, which represent cases in which positive samples are incorrectly labelled as negative. Even though the percentage of false negatives is small, it indicates situations in which the model is unable to detect positive cases.

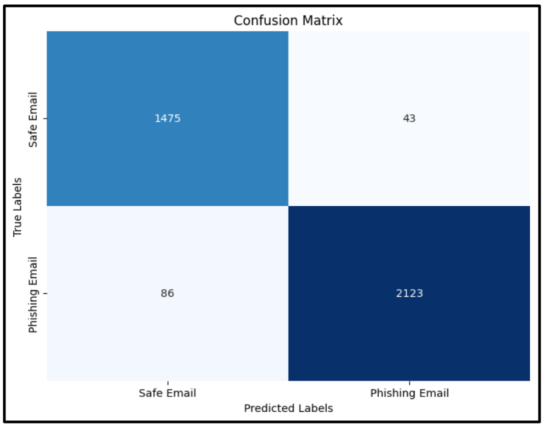

Figure 7. RCNN Model Confusion Matrix

According to Confusion Matrix in Figure 7 the evaluation metrics for model testing calculated for each class of the data as shown in Table 3.

Table 3. RCNN Classification Report

Class	Accuracy	Precision	Recall	F-Score
Safe Mail	0.97	0.95	0.97	0.96
Phishing Mail	0.97	0.98	0.96	0.97
Macro Avg	0.97	0.96	0.97	0.97
Weighted Avg	0.97	0.97	0.97	0.97

TIME DISCUSSION

The time taken to train and test different sorts of algorithms is effective in estimating the computational capacity of the algorithms and their ability to solve real-time problems. For the CNN, the training phase takes a lot of time while during the inference phase it is very efficient. This makes it ideal for occurrences where the speed of the decision made is very paramount. However, the CNN-RNN model which incorporates Recurrent Neural Network into CNN entails a longer time of training as well as a way longer time of testing. This implies that, the combination of a more complex architecture, by the inclusion of the RNN component adds more computational demand in both training and inference. Therefore, even though the CNN-RNN may present more efficient results in specific tasks dealing with sequence data, the method is less proficient when it comes to real-time applications testing time.

The proposed Region-based Convolutional Neural Network (RCNN) also has a moderate training and testing times. This model that is suitable for tasks like object detection is a reasonable middle-ground, allowing faster inference while at the same time increasing detection accuracy.

To sum up, the CNN is the most efficient one when it comes to the real-time applications as its testing time is in this case the shortest one. The CNN-RNN with its extensions on the other hand, offers enhanced capabilities at the cost of a little slower inference time in the sequential problems. In this case, the RCNN can be highlighted as a balanced one since it provides the detected objects with adequate accuracy and at an acceptable time, which is crucial in many cases.

Table 4. Time Analysis

Algorithm	Training Time (S)	Testing Time (S)
CNN	2026	1.01
CNN-RNN	2277	5.042
RCNN	2080	2.014

CONCLUSION

In the context of binary classification tasks using sequential data, we comprehensively investigated the performance of several sequential models in this study, including CNN, CNN-RNN, RCNN. Uneven labelling in the 'Email Type' column of the dataset—13,055 samples for training and 5,595 samples for testing—posed a problem. To handle missing values, tokenize and pad text input, and label- encode the target variables, the pretreatment stages were carefully carried out. These actions established the framework for the machine learning models' evaluation and training. The implemented algorithms displayed a wide range of topologies, each intended to capture a distinct facet of the input data's sequential relationships.

To evaluate the performance of the models, evaluation metrics such as Accuracy, Precision, Recall, and F1-Score were utilized. An in-depth examination of confusion matrices shed light on how well the models differentiated between positive and negative examples. All algorithms performed well, regularly attaining metrics above 94%; nevertheless, Top performances were, CNN, and RCNN, with 97% of all evaluation measures met. The study also highlighted the applicability of models such as CNN-RNN. which consistently yielded well-balanced and high-performing results.

With a variety of model options depending on particular needs and priorities, these findings offer insightful information to academics and practitioners working on related projects. This article presents a thorough evaluation methodology that advances our knowledge of model behavior in the context of binary classification using sequential data that is imbalanced. Subsequent investigations may examine additional enhancements and expansions to tackle obstacles in practical uses.

REFERENCES

1] K. Cabaj, D. Domingos, Z. Kotulski, and A. Respício,—Cybersecurity education: Evolution of the discipline and analysis of master programs,‖ Comput. Secur., vol. 75, pp. 24–35, 2018, doi: 10.1016/j.cose.2018.01.015.

[2] C. Iwendi et al., —KeySplitWatermark: Zero Watermarking Algorithm for Software Protection against Cyber-Attacks,‖ IEEE Access, vol. 8, pp. 72650–72660, 2020, doi:10.1109/ACCESS.2020.2988160.

[3] A. R. Javed, M. Usman, S. U. Rehman, M. U. Khan, and M. S. Haghighi, —Anomaly Detection in Automated Vehicles Using Multistage Attention-Based Convolutional Neural Network,‖ IEEE Trans. Intell. Transp. Syst., vol. 22, no. 7, pp. 4291–4300, 2021, doi: 10.1109/TITS.2020.3025875.

[4] M. Mittal, C. Iwendi, S. Khan, and A. R. Javed,—Analysis of security and energy efficiency for shortest route discovery in low- energy adaptive clustering hierarchy protocol using Levenberg- Marquardt neural network and gated recurrent unit for intrusion detection system,‖ Trans. Emerg. Telecommun. Technol., vol. 32, 2020, [Online]. Available: https://api.semanticscholar.org/CorpusID:219918712

[5] G. Aaron, —Phishing Activity Trends Report 2nd Quarter,‖ Anti-Phishing Work. Gr., no. September, pp. 1–12, 2019, [Online]. Available: https://apwg.org/trendsreports/

[6] V. Zeng, S. Baki, A. El Aassal, R. Verma, L. F. T. De Moraes, and A. Das, —Diverse datasets and a customizable benchmarking framework for phishing,‖ IWSPA 2020 - Proc. 6th Int. Work. Secur. Priv. Anal., no. Section 3, pp. 35–41, 2020, doi: 10.1145/3375708.3380313.

[7] A. Basit, M. Zafar, X. Liu, A. R. Javed, Z. Jalil, and K. Kifayat, —A comprehensive survey of AI-enabled phishing attacks detection techniques,‖ Telecommun. Syst., vol. 76, no. 1, pp. 139–154, 2021, doi: 10.1007/s11235-020-00733-2.

[8] N. Moradpoor, B. Clavie, and B. Buchanan,—Employing machine learning techniques for detection and classification of phishing emails,‖ Proc. Comput. Conf. 2017, vol. 2018-Janua, no. July, pp. 149–156, 2018, doi: 10.1109/SAI.2017.8252096.

[9] C. S. Jalda, A. Kumar Nanda, and R. Pitchai,—Spoofing E-Mail Detection Using Stacking Algorithm,‖ in 2022 8th International Conference on Smart Structures and Systems (ICSSS), 2022, pp. 1–4. doi: 10.1109/ICSSS54381.2022.9782173.

[10] H. Abroshan, J. Devos, G. Poels, and E. Laermans,—Phishing Happens beyond Technology: The Effects of Human Behaviors and Demographics on Each Step of a Phishing Process,‖ IEEE Access, vol. 9, pp. 44928– 44949, 2021, doi: 10.1109/ACCESS.2021.3066383.

[11] N. Q. Do, A. Selamat, O. Krejcar, E. Herrera-Viedma, and H. Fujita, —Deep Learning for Phishing Detection: Taxonomy, Current Challenges and Future Directions,‖ IEEE Access, vol. 10, pp. 36429–36463, 2022, doi: 10.1109/ACCESS.2022.3151903.

[12] S. Bagui, D. Nandi, S. Bagui, and R. J. White,—Machine Learning and Deep Learning for Phishing Email Classification using One-Hot Encoding,‖ J. Comput. Sci., vol. 17, no. 7, pp. 610–623, 2021, doi: 10.3844/jcssp.2021.610.623.

[13] M. Dewis and T. Viana, —Phish Responder: A Hybrid Machine Learning Approach to Detect Phishing and Spam Emails,‖ Appl. Syst. Innov., vol. 5, no. 4, pp. 0–1, 2022, doi: 10.3390/asi5040073.

[14] A. Mughaid, S. AlZu'bi, A. Hnaif, S. Taamneh, A. Alnajjar, and E. A. Elsoud, —An intelligent cyber security phishing detection system using deep learning techniques,‖ Cluster Comput., vol. 25, no. 6, pp. 3819–3828, 2022, doi: 10.1007/s10586-022-03604-4.

[15] U. A. Butt, R. Amin, H. Aldabbas, S. Mohan, B. Alouffi, and A. Ahmadian, —Cloud-based email phishing attack using machine and deep learning algorithm,‖ Complex Intell. Syst., vol. 9, no. 3, pp. 3043–3070, 2023, doi: 10.1007/s40747-022-00760-3.

[16] Q. Qi, Z. Wang, Y. Xu, Y. Fang, and C. Wang,—Enhancing Phishing Email Detection through Ensemble Learning and Undersampling,‖ Appl. Sci., vol. 13, no. 15, 2023, doi: 10.3390/app13158756.

[17] Y. S. Murti and P. Naveen, —Machine Learning Algorithms for Phishing Email Detection,‖ J. Logist. Informatics Serv. Sci., vol. 10, no. 2, pp. 249–261,2023, doi: 10.33168/JLISS.2023.0217.

[18] M. J. Keelan Evans, Alsharif Abuadbba, Tingmin Wu, Kristen Moore, Mohiuddin Ahmed, Ganna Pogrebna, Surya Nepal, —RAIDER: Reinforcement-aided Spear Phishing Detector,‖ arXiv:2105.07582v3, no. 1, pp. 1– 17, 2023.

[19] M. Dewis and T. Viana, —Phish Responder: A Hybrid Machine Learning Approach to Detect Phishing and Spam Emails,‖ Appl. Syst. Innov., vol. 5, no. 4, 2022, doi: 10.3390/asi5040073.

[20] A. Alhogail and A. Alsabih, —Applying machine learning and natural language processing to detect phishing email,‖ Comput. Secur., vol. 110, p. 102414, 2021, doi: https://doi.org/10.1016/j.cose.2021.102414.

[21] S. A. A. Ghaleb, M. Mohamad, S. A. Fadzli, and W. A.H. M. Ghanem, —Training Neural Networks by Enhance Grasshopper Optimization Algorithm for Spam Detection System,‖ IEEE Access, vol. 9, pp. 116768–116813, 2021, doi:10.1109/ACCESS.2021.3105914.

[22] R. Eckhardt and S. Bagui, —Convolutional Neural Networks and Long Short Term Memory for Phishing Email Classification,‖ Int. J. if Comput. Sci. Inf. Secur., vol. 19, no. 5, pp. 27–35, 2021.

[23] Ž. Vujović, —Classification Model Evaluation Metrics,‖ Int. J. Adv. Comput. Sci. Appl., vol. 12, no. 6, pp. 599– 606, 2021, doi: 10.14569/IJACSA.2021.0120670.