FUDMA Journal of Engineering and Technology
ISSN: 3092-9385
https://sites.google.com/fudutsinma.edu.ng/fjet
Volume 1, Issue 2, 2025, 35-47

Short Messaging Service Spam Detection Model Using Natural Language Processing and Deep Learning Techniques

Olatunde A. AKANO[1], Wariz A. ISMAEL[2], Ayomikun A. AWOSEYI[3*], Femi AYO[4], Ifeoluwa M. OLANIYI[5], Jide E.T. AKINSOLA[6]

[1,2,3,5,6]Department of Computer Sciences, Abiola Ajimobi, Technical University, Ibadan, Oyo State, Nigeria
[4]Department of Mathematical Sciences, Olabisi Onabanjo University, Ago-Iwoye, Ogun State, Nigeria

[1]olatunde.akano@tech-u.edu.ng, [2]warizakorede64@gmail.com, [3*]awoseyiayomikun@gmail.com, [4]ayo.femi@oouagoiwoye.edu.ng, [5]olaniyiifeoluwa12@gmail.com, [6]akinsolajet@gmail.com

Abstract

Unsolicited Short Message Service (SMS) messages, or SMS spams, pose a major challenge in mobile communication. These unwanted messages compromise user privacy, leading to data bridge or financial risks. To address this growing concern, this study explores the implementation of deep learning and Natural Language Processing (NLP) procedures to effectively detect SMS spam. By developing a robust spam detection system, this study enhances the security and usability of mobile communication platforms. This study implements an effective spam detection system using deep learning and NLP techniques. The system was developed using Python 3.10 within the Google Collaboratory environment. The SMS Spam Collection dataset, consisting of 5,574 characterized messages, underwent preprocessing procedures that included tokenization, stopword removal, lemmatization, and transformation using Term Frequency-Inverse Document Frequency (TF-IDF) vectorization. Three deep learning models were implemented for classification: Convolutional Neural Networks (CNN), Long Short-Term Memory networks (LSTM), and Recurrent Neural Networks (RNN). These models were trained and evaluated using performance metrics such as correctness, precision, recall, F1-score, Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE). Among the models tested, the CNN model demonstrated the best performance, achieving an accuracy of 96.90 percent, a precision of 0.9692, a recall of 0.9690, and an F1-score of 0.9691. It also had the lowest error rates, indicating its superior predictive capability. The results confirm the effectiveness of CNNs for SMS spam detection, particularly when combined with rigorous text preprocessing. The study suggests for further study, the application of federated leaning for modelling SMS spam detection.

Keywords: *Deep learning, machine learning, natural language processing, short message service, SMS spam.*

1.0 Introduction

Short Message Service (SMS) is a widely used text messaging protocol that has become an integral component of modern mobile communication. It was initially developed for sending short, text-based messages between mobile devices. SMS is now used by individuals and businesses alike for a range of purposes including personal communication, business promotions, customer engagement, and notifications [1]. Its high accessibility, ease of use, and compatibility across mobile networks has made SMS an essential medium for reaching users globally. As businesses increasingly rely on SMS to connect with their customers, the need for secure and reliable SMS communication becomes even more critical [2].

Spam messages impact user experience, compromise privacy, and may lead to security risks. Traditional methods fall short against increasingly sophisticated spam techniques, creating a demand for automated spam detection [3]. The upsurge in SMS adoption has been accompanied by corresponding rise in spam messages, creating significant setbacks for operators and mobile carriers [4]. SMS spam messages, which include unsolicited advertising, phishing attempts, and deceptive offers, can intrude on user privacy, lead to financial fraud, and waste network resources [5]. Unlike email spam, SMS spam is harder to filter due to the brevity of messages, limited available datasets, and informal language used in texting [6]. Traditional spam detection techniques often fall short in tackling the sophisticated tactics employed by spammers today. Consequently, there is a growing request for automatic SMS spam recognition systems capable of efficiently filtering out spam messages to improve user experience and communication security [7].

Machine learning and Natural Language Processing (NLP) provide effective tools to combat SMS spam, leveraging algorithms like Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), and Long Short-Term Memory (LSTM) for enhanced spam classification. Recent research has increasingly emphasized on machine learning and NLP methods to lessen the encounters modeled by SMS spam [8]. Deep learning models, including CNN, RNN, and LSTM networks, offer powerful methods for spam classification by repeatedly mining features from text data [9]. These models are predominantly suitable for tasks related to progressive and context-

Received: 01/07/2025; Revised: 05/07/2025; Accepted: 05/07/2025; Published: 08/08/2025

dependent data, for instance, SMS text messages. In this study, CNN, RNN, and LSTM algorithms were applied to develop a reliable spam detection system and categorize the most suitable model required for SMS spam classification. The CNN architecture, in particular, leveraged the pooling theorem to diminish the dimensionality of feature maps while retaining the most critical features, enhancing computational efficiency and improving classification accuracy [10]. The study aims to create a SMS spam detection system using NLP techniques for data preprocessing and deep learning models to improve spam classification accuracy.

According to [11], spam filtering techniques for SMS data focus on the use of LSTM and RNN networks to enhance memory cell storage and control information flow. The application of machine learning and deep learning means effectively reduces error rates through feature extraction and algorithm evaluation. However, the existing system suffers from low accuracy, poor dataset selection, and inadequate feature extraction, resulting in a complex and less comprehensible structure [12]. Additionally, the study lacks efforts to address bias reduction and optimize advanced feature extraction techniques. [13] highlights a study that focused on integrating data preprocessing, embedding, and classification modeling, leveraging autoencoders and hybrid models to improve spam classification accuracy. Its innovative aggregation of machine learning and deep learning techniques proved effective but lacked an exploration of practical challenges and limitations in real-world applications. Additionally, the research did not compare the proposed models and techniques with advanced methods in spam discovery, highlighting a gap in benchmarking and optimization.

[14] utilized deep learning models, including RNN, LSTM, and CNN, to address multilingual spam detection across datasets in English, Korean, and mixed languages. Its strength lies in its comprehensive and sophisticated methodology, ensuring accurate classification through diverse datasets. However, the study's focus on image-based processing overlooked the nuances of text-based spam detection, limiting its applicability. It also acknowledged the difficulty of applying current detection models to evolving spam text formats, suggesting the need for adaptable solutions [15]. As discussed by [16], a structured approach to spam filtering using datasets labeled with spam and ham messages. Preprocessing techniques such as stopword removal, tokenization, and attribute selection prepared the data for advanced algorithm training and testing. While the methodology was effective, challenges arose due to limited message size and incomplete information, impacting accuracy. The study also raised concerns about the broader implications of digital platforms on self-expression and cognitive abilities, emphasizing the complexities in adapting spam filtering systems to modern communication patterns.

According to [17], a comprehensive methodology for spam appraisal discovery comprising four levels: data aggregation and preprocessing using NLP techniques, vigorous learning to label unlabeled data, feature selection with methods like TF-IDF and word embeddings, and spam discovery using traditional machine learning (SVM, KNN, Naive Bayes) and deep learning classifiers (MLP, CNN, LSTM). The integration of traditional and deep learning methods resulted in high classification accuracy. However, the study faced limitations due to small dataset sizes, such as the Ott dataset with only 1,600 reviews, and inadequate attention to addressing review spammers, indicating a need for larger datasets and broader considerations in future research. According to [18], two open-source email datasets were used, with preprocessing steps like removing stop words and punctuation marks before automatically extracting features during model training. Advanced techniques, including BiLSTM and BERT, were employed, with BERT achieving superior performance in spam email detection. Despite its effectiveness, the study lacked a detailed discussion on model limitations and dataset biases, leaving room for further research to explore the generalizability of the approach to other NLP tasks and the robustness of the models against evolving spam email tactics.

[19] focused on SMS spam classification using a combination of traditional and deep learning algorithms, including SVM, KNN, DT, RF, and hybrid CNN-LSTM models, with preprocessing steps like TF-IDF and word embeddings to optimize data representation. The study demonstrated a meticulous approach to text classification and achieved promising results. However, it lacked a detailed discussion on the specific parameters and hyperparameters used, limiting reproducibility and generalizability. Additionally, it offered limited exploration of feature transformation techniques and their influence on model performance, presenting opportunities for refinement in future studies [20]. [2] evaluated SMS spam classification using two datasets, ExAIS_SMS and UCI, with preprocessing steps that removed duplicates and native languages. The BiLSTM model, implemented using MATLAB and WEKA, demonstrated significant performance improvements over traditional machine learning algorithms measured on metrics like precision, recall, F-measure, and accuracy. While the study effectively showcased the strengths of deep learning in natural language-based applications, it lacked a detailed exploration of feature engineering and the impact of preprocessing methods on model performance. Further testing on diverse datasets was recommended to enhance the scalability and generalizability of the BiLSTM model [21] .

According to [3], the study employed word integration techniques such as TF-IDF, Word2vec, and GloVe alongside preprocessing methods like tokenization, stopword removal, and lemmatization for SMS spam classification. Using UCI-sourced datasets, the research highlighted the importance of the BERT model to attain high performance through contextual sentence embedding. Despite its comprehensive approach, the study faced limitations in generalizing findings due to its reliance on specific datasets. It also lacked an exploration of ensemble

Received: 01/07/2025; Revised: 05/07/2025; Accepted: 05/07/2025; Published: 08/08/2025

learning and hybrid models that could combine machine learning and deep learning methods for improved characterization results. [22] analyzes text messages using deep learning to discriminate spam from non-spam. More specifically, LSTM and CNN were used. The feature set of the suggested models was self-extracted and was only dependent on text input. [23] introduces a mobile application that detects and prevents smishing attacks by utilizing a rule-based SMS service. More precisely, the created SMS service enables the SMS mobile application to seize SMS messages en route to a mobile device. The seized messages were subsequently distributed to the rule-based machine learning model through an Application Programming Interface (API). By applying the meticulously chosen rules to the retrieved message, the model determines whether the message is spam or a ruse. The outcome of the evaluation is subsequently transmitted to the mobile application via the API.

[24] focused on binary classification of spam and ham SMS messages using datasets which source is the UCI Machine Learning database. It combined advanced feature selection with a Stacked Restricted Boltzmann Machine (RBM) and a Deep Neural Network (DNN) classifier to create a robust detection system. The methodology ensured high accuracy but introduced complexity due to the specialized algorithms used, which required advanced knowledge and resources for implementation. Additionally, the study lacked a detailed discussion of performance evaluation metrics, which limited intuitions into the system's overall effectiveness [25]. [26] proposed a hybrid deep learning model for email spam characterization using a fuzzy inference system. The methodology involved collecting datasets from Enron and Spam Assassin, applying a hybrid feature selection approach, and using an ensemble of classifiers to improve accuracy. While the approach demonstrated strong feature selection and adaptability, it faced limitations such as potential overfitting, reliance on specific datasets, and high computational complexity. Additionally, it lacked real-time learning mechanisms, making it less effective in adapting to evolving spam techniques like phishing and social engineering attacks. Many studies lacked robust feature selection techniques, leading to reduced classification accuracy. Some methods also struggled with data imbalance issues, impacting performance. Additionally, insufficient preprocessing in certain studies resulted in increased complexity. These gaps highlight the need for improved preprocessing, effective feature selection, and advanced classifiers to enhance spam detection accuracy.

2.0 Materials and Methods

This section summarizes the methods, techniques, and tools employed in the development and implementation of the SMS spam detection system. It consists of a detailed elucidation of the dataset, preprocessing techniques, the system architecture, and the training process for the models used. The proposed system architecture, as described in Figure 1, entails four stages. First, the Dataset serves as the input, consisting of an SMS message dataset. Next is Data Preprocessing, which encompasses Data Cleaning (such as lowercasing and punctuation removal) and NLP Pre-Processing techniques, including tokenization, stopword removal, stemming/lemmatization, word embedding, and TF-IDF transformation. The preprocessed data is then inputted into the Model, which utilizes text-based deep learning architectures for example, CNN, LSTM, and RNN to process the data. Finally, the system performs Classification, categorizing messages as either SPAM or NOT SPAM.

2.1 Dataset

The dataset used in this study is the SMS Spam Collection Dataset, fetched from the Kaggle database. In this dataset, there are 5,574 SMS messages in English, characterized as either spam or legitimate. Each message is categorized in two columns: v1, which specifies the tag (ham or spam), and v2, which holds the unprocessed writing. The dataset is obtained from various platforms to ensure diversity and comprehensiveness, providing a robust foundation for training and evaluating spam detection models. The sources and corresponding message counts are summarized in Table 1.

Received: 01/07/2025; Revised: 05/07/2025; Accepted: 05/07/2025; Published: 08/08/2025

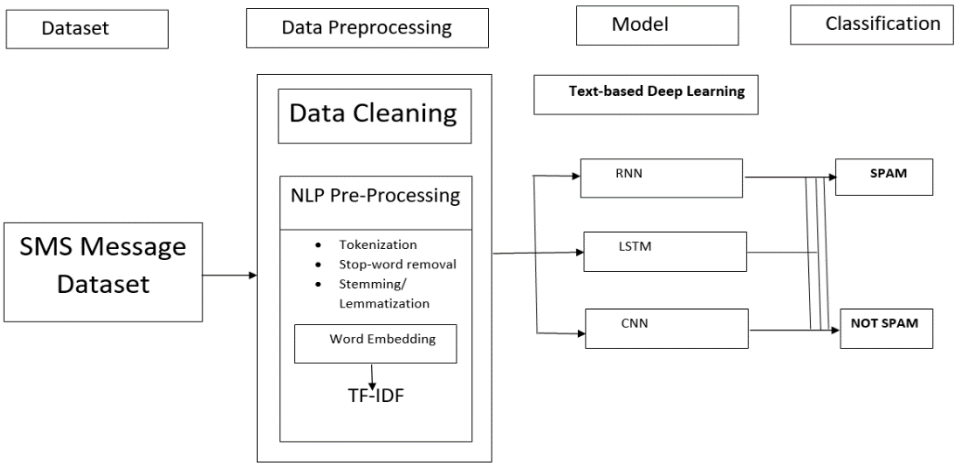

Figure 1: SMS Spam detection architectural framework

Table 1: Details of the dataset:

Sources	Total Messages
Grumbletext website	425
NUS SMS Corpus (NSC).	3375
Caroline Tag's PhD thesis	450
Spam Corpus v.0.1 Big	1324
Total	**5574**

This dataset comprises 747 junk messages and 4,825 ham text messages, representing a variety of real-world scenarios for SMS communication. The diverse sources ensure exposure to massive spam and legitimate SMS circumstances, enhancing the generalizability of the prepared models.

The inclusion of multiple sources makes this dataset suitable for research in SMS spam detection, offering balanced data and representative patterns for effective spam classification.

2.2 Data Processing

Data preprocessing is a significant stage in preparing the dataset for deep learning models. The goal is to translate raw text data into a numerical state such that the algorithms can process and understand. The preprocessing pipeline includes data cleaning and the application of several NLP procedures.

a. Data Cleaning

Data cleaning involves removing irrelevant components from the text to improve the model's performance. Key steps contain:

i. **Capitalization Removal**: All text is converted to lowercase to ensure uniformity.
ii. **Punctuation Removal**: Unnecessary symbols and punctuation marks are eliminated to simplify the text structure.

b. NLP Implementation

NLP procedures are used to extract meaningful features from the text, ensuring it is suitable for machine learning. The techniques applied include:

i. **Tokenization**: Tokenization is the method of reducing complex text into smaller uncomplicated text known as tokens. Words or phrases are signified by these tokens, facilitating analysis. In this study, the Natural Language Toolkit (NLTK) library was adopted to tokenize the SMS messages. Tokenization aids in understanding text sequences and is foundational to text preprocessing.
ii. **Stopword Removal**: Stopwords are common words such as *"the," "is,"* and *"at"* that transmit smaller amount of semantic value. In order to ensure noise reduction and improved efficiency of the dataset, these words are removed. This step guarantees the model's emphases on the most informative words in the text.
iii. **Lemmatization**: Lemmatization reverses words to their original or root form, ensuring uniformity across diverse word forms. For instance, words like *"working," "worked,"* and *"works"* are normalized to *"work."* This process advances the simplification of the model by treating related words as identical.
iv. **TF-IDF Vectorization**: Term Frequency-Inverse Document Frequency (TF-IDF) is used to assign weights to words based on their importance within a document and across the entire corpus. Words that frequently appear in a single SMS but rarely across all messages are given higher weights, as they

Received: 01/07/2025; Revised: 05/07/2025; Accepted: 05/07/2025; Published: 08/08/2025

are more likely to indicate spam. The text was converted into a numerical format by means of TF-IDF vectorization, which captures the importance of words by assigning weights based on their frequency in the message and their rarity across the entire dataset. The number of features was limited to 3000 using the *max_features* parameter to reduce dimensionality.

TF-IDF is calculated using the following formulas:

$$TF - IDF(t, d, D) = TF(t, d) \times IDF(t, D) \quad (1)$$

where:

$TF(t, d)$, is the relative frequency of term t within document d,

$$TF(t, d) = \frac{f_{t,d}}{\sum_{t' \epsilon d} \quad f_{t',d}} \quad (2)$$

$f_{t,d}$ is the root amount of the term t in document d, and the denominator is the total count of all terms in d.
$IDF(t, D)$: Inverse Document Frequency, which quantifies how rare a term is across the corpus D.

$$IDF(t, D) = log \frac{N}{|\{d: t \epsilon d\}|} \quad (3)$$

where N is aggregate frequency of documents in the corpus $|\{d: t \epsilon d\}|$: is the number containing the term t.

This comprehensive preprocessing pipeline ensures that the dataset is optimally prepared for deep learning model training, enabling accurate and efficient SMS spam detection.

2.3 Model Training

This section focuses on building and evaluating deep learning (DL) models for detecting SMS spam messages. To determine the most effective classification algorithm, dataset splitting ratios, such as 90:10, 80:20, and 70:30, were used to generate performance evaluation metrics for various deep learning models. The dataset was split into training and testing sets using these ratios, with the classifier that yielded the best performance selected for further development.

The system leverages Python libraries such as Scikit-learn, NumPy, pandas, NLTK, Keras, and TensorFlow. The model was trained across different random states to ensure the best possible accuracy value, and the best-performing algorithm was chosen as the golden model for SMS spam detection. Three deep learning models were used in this study: Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs), and Long Short-Term Memory (LSTM) networks.

2.3.1 CNN Model

The convolutional layer in a CNN is primarily responsible for detecting significant features within the text data. The convolution technique is applied to the word vectors created by the Word Surrounding layer. Let x_i represent the d-dimensional word vector associated with the $i - th$ word in the message, where $x_i \in R^d$. The input message is represented as $x \in R^{L \times d}$, where L is the length of the message. A window vector w_j is defined to consist of k consecutive word vectors, and the convolution operation produces a feature map $c \in R^{L-k+1}$. The feature map is computed by applying a filter $p \in R^{k \times d}$ to the window, using the formula:

$$c_j = f\left(w_j \odot p + b\right) \quad (4)$$

where f is a nonlinear function (ReLU), and $\odot$ denotes the element-wise product.

The CNN model was implemented to detect the local features of the input sequences. It consisted of the following layers:

Input Layer: The TF-IDF vectors were fed into the CNN as input.

Convolutional Layer: A 1D convolutional layer was used to extract features from the TF-IDF vectors.

ReLU Activation: The Rectified Linear Unit (ReLU) activation function was adopted to introduce non-linearity.

Max Pooling: A max-pooling layer was introduced to down-sample the feature maps and reduces dimensionality.

Fully Connected Layer: The output from the pooling layer was passed through a fully connected layer for final classification.

Received: 01/07/2025; Revised: 05/07/2025; Accepted: 05/07/2025; Published: 08/08/2025

2.3.2 LSTM Training

LSTM is a variant of RNN intended to handle lasting dependencies in sequences. The LSTM architecture comprises multiple gates: the input gate i_t, the output gate o_t, and the forget gate f_t, which regulates the stream of data through the model. The LSTM unit's transition functions at time step t are defined as:

$$i_t = \sigma(W_i.[h_{t-1} + b_i]) \tag{5}$$

$$f_t = \sigma\big(W_f.[h_{t-1}, x_t] + b_f\big) \tag{6}$$

$$q_t = tanh\big(W_q.[h_{t-1}, x_t] + b_q\big) \tag{7}$$

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o) \tag{8}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot q_t \tag{9}$$

$$h_t = o_t \odot tanh(c_t) \tag{10}$$

where x_t is the input vector, bσ is the sigmoid function, and $tanh$ is the hyperbolic tangent function. The weight matrices W and bias vectors b are learned in the process of training.

LSTM networks were employed to model the sequential nature of SMS messages. The architecture included:

Input Layer: The TF-IDF vectors were used as input instead of word embeddings.

LSTM Layer: This layer captured the sequential dependencies in the data using a fixed number of LSTM units.

Dropout Layer: A dropout layer was included to mitigate overfitting while training.

Dense Layer: The concluding dense layer together with a sigmoid activation function was used for binary characterization.

2.3.3 RNN Training

RNN are designed to develop chronological data by utilizing the output from the preceding time step as the input for the recent time step. The concealed state h_t in an RNN stores the memory of previous inputs and is updated with each time step. The recurrence relations are as follows:

$$h_t = f(h_{t-1}, x_t) \tag{11}$$

$$h_t = tanh\big(W_{hh}h_{t-1} + W_{xh}X_t\big) \tag{12}$$

$$y_t = W_{hy}h_t \tag{13}$$

where W represents the weight matrices, $tanh$ is the activation function, y_t is the outcome, and h_t represent the current state.

RNNs were used to handle sequential data where the output from the previous time step is fed into the current time step, enabling the model to remember prior context. The RNN model was structured as follows:

Embedding Layer: Word embedding was used to represent the text in a dense format.

RNN Layer: The RNN layer consisted of units that allowed the network to maintain information about previous words in the sequence.

Dense Layer: A fully associated layer was adopted at the end for classification with a sigmoid activation function to decide whether a message is spam or not.

2.3.4 Training of the Models

For training, the following steps were followed:

Optimization: The Adam optimizer was implemented because of its proficiency in conducting sparse gradients and its ability to adapt the learning rate during training.

Loss Function: Binary cross-entropy was used as the loss function since this is a binary characterization task (spam vs. ham).

Evaluation Metrics: Accuracy, precision, recall, F1-score, mean squared error (MSE), root mean squared error (RMSE), and mean absolute error (MAE) were introduced to estimate the concert of the models.

Received: 01/07/2025; Revised: 05/07/2025; Accepted: 05/07/2025; Published: 08/08/2025

Training Epochs: The models were trained for a fixed number of epochs (e.g., 10-20), with early stopping criteria to prevent overfitting.

The models were implemented using popular Python libraries such as **TensorFlow**, **Keras**, and **scikit-learn**, which provided the necessary tools for building and training the models efficiently.

2.4 Evaluation Metrics

The following metrics were adopted to measure the models' concert:

i. **Accuracy**: Accuracy measures the proportion of correctly classified messages (spam or ham) out of the total messages:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{14}$$

where TP represent True positive, TN represent True Negative, FP denotes False Positive, and FN denotes False Negative.

ii. **Precision**: Precision quantifies the percentage of correctly classified spam messages out of all messages classified as spam:

$$Precision = \frac{TP}{TP+FP} \tag{15}$$

where TP represents True positive, and FP denotes False Positive.

iii. **Recall (Sensitivity)**: Recall indicates the percentage of actual spam messages correctly recognized by the model

iv. **F1-Score**: F1-**score** is the harmonic mean of precision and recall, providing a balanced measure of a model's performance:

$$F1-score = \frac{2\times(Precision\times Recall)}{Precision+Recall} \tag{16}$$

v. **Mean Squared Error (MSE)**: MSE measures the average squared difference between predicted and actual values, representing the overall prediction error:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{17}$$

where n is the quantity of fitted point, y_i is the prediction and $\hat{y}_i$ is the true value.

vi. **Root Mean Squared Error (RMSE)**: RMSE is the square root of MSE, providing a measure of the average magnitude of prediction errors:

$$RMSE = \sqrt{MSE} \tag{18}$$

vii. **Mean Absolute Error (MAE)**: MAE computes the average of the absolute differences between predicted and actual values:

$$MAE = \frac{1}{n}\sum_{i-1}^{n}|y_i - \hat{y}_i|^2 \tag{19}$$

where n is the quantity of fitted point, y_i is the prediction and $\hat{y}_i$ is the true value.

These metrics provide a comprehensive assessment of the models' classification performance.

2.5 Architectures of the Models

All models were implemented using the Keras API with TensorFlow as backend and trained using Python 3.10 in the Google Collaboratory environment. Each model utilized an embedding layer as input and terminated with a dense layer for binary classification using a softmax activation function.

2.5.1 CNN Architecture

The CNN model captures local word patterns and is optimized for performance and simplicity. It includes the following layers:

Embedding Layer: Vocabulary size = 5000, output dimension = 128, input length = padded sequence length.

Received: 01/07/2025; Revised: 05/07/2025; Accepted: 05/07/2025; Published: 08/08/2025

SpatialDropout1D Layer: Dropout rate = 0.2 to reduce overfitting.
Conv1D Layer: 128 filters, kernel size = 5, ReLU activation.
GlobalMaxPooling1D Layer: Extracts the most important features from each feature map.
Dense Layer: Output layer with 2 neurons and softmax activation for binary classification.

2.5.2 LSTM Architecture

The LSTM model is designed to retain sequential dependencies and includes:
Embedding Layer: Same configuration as CNN.
SpatialDropout1D Layer: Dropout rate = 0.2.
Layer: 100 memory units, dropout = 0.2, recurrent dropout = 0.2.
Dense Layer: 2 neurons with softmax activation.

2.5.3 RNN Architecture

The RNN model serves as a simpler recurrent baseline with:
Embedding Layer: Same configuration as others.
SpatialDropout1D Layer: Dropout rate = 0.2.
SimpleRNN Layer: 100 units with default tanh activation.
Dense Layer: 2 neurons with softmax activation.

3.0 Results and Discussion

This segment explains the outcomes of the deep learning models for SMS spam recognition and discusses their performance. The models evaluated include CNN, LSTM, and RNN. Performance measures such as accuracy, precision, recall, F1-score, MSE, RMSE, and MAE were used to quantify the models' effectiveness.

3.1 Models' Performance Evaluation

The performance evaluation of the models is elucidated accordingly.

3.1.1 CNN Performance Results

The CNN model had the most performance among the tested models. As shown in Table 2, the CNN attained an accuracy of 96.90%, a precision of 0.9692, recall of 0.9690, and an F1-score of 0.9691. These results highlight the model's ability to balance precision and recall, diminishing false positives and false negatives. Furthermore, the error metrics indicate that the CNN has the lowest MSE (0.0309), RMSE (0.1759), and MAE (0.0309), demonstrating its superior capability in reducing prediction errors. It is worthy of note that, the lower cost metrics such as MSE, MAE and RSME, the better and more reliable the model. Hence, the values that tends to zero are of good significance.

Table 2: Performance metrics for CNN

S/N	Metrics	Result
1.	Accuracy	96.90%
2.	Precision	0.9692
3.	Recall	0.9690
4.	F1-score	0.9691
5.	MSE	0.0309
6.	RMSE	0.1759
7.	MAE	0.0309

3.1.2 LSTM Performance Results

The LSTM model also executed well, achieving an accuracy of 96.13% and an F1-score of 0.9624, as summarized in Table 3. However, its error metrics (MSE: 0.0386, RMSE: 0.1966, MAE: 0.0386) were slightly higher than those of the CNN, indicating a higher likelihood of classification errors. Despite these differences, the LSTM demonstrated strong overall performance, particularly in handling sequential dependencies.

Table 3: Performance metrics for LSTM

S/N	Metrics	Result
1.	Accuracy	96.13%
2.	Precision	0.9650
3.	Recall	0.9613
4.	F1-score	0.9624

Received: 01/07/2025; Revised: 05/07/2025; Accepted: 05/07/2025; Published: 08/08/2025

S/N	Metrics	Result
5.	MSE	0.0386
6.	RMSE	0.1966
7.	MAE	0.0386

3.1.3 RNN Performance Results

The RNN model attained an accuracy of 95.93%, as explained in Table 4. Its precision (0.9636), recall (0.9593), and F1-score (0.9606) were a little lesser than those of the LSTM and CNN models. Similarly, the RNN's error metrics (MSE: 0.0406, RMSE: 0.2015, MAE: 0.0406) were the highest among the three models, reflecting its relatively weaker performance.

Table 4: Performance metrics for RNN

S/N	Metrics	Result
1.	Accuracy	95.93%
2.	Precision	0.9636
3.	Recall	0.9593
4.	F1-score	0.9606
5.	MSE	0.0406
6.	RMSE	0.2015
7.	MAE	0.0406

3.2 Discussion of Results

This section discusses the performance evaluation results accordingly.

3.2.1 Comparative Analysis of Deep Learning Algorithms

The comparative analysis of the CNN, LSTM, and RNN models highlights the CNN as the best-performing model across all metrics. As revealed in Table 5, the CNN achieves the peak accuracy (96.90%) and the lowest error rates (MSE: 0.0309, RMSE: 0.1759, MAE: 0.0309), demonstrating its superior ability to classify SMS messages accurately with minimal errors.

The LSTM model ranks second, with an accuracy of 96.13% and error metrics slightly higher than the CNN. Its sequential processing capabilities provided strong results, particularly in capturing dependencies within the dataset. However, the RNN model, while effective, showed the lowest performance in terms of accuracy (95.93%) and the highest error rates.

Table 5: Comparative metrics for deep learning models

Algorithms	Accuracy	Precision	Recall	F1-Score	MSE	RMSE	MAE
CNN	96.90%	0.9692	0.9690	0.9691	0.0309	0.1759	0.0309
LSTM	96.13%	0.9650	0.9613	0.9624	0.0386	0.1966	0.0386
RNN	95.93%	0.9636	0.9593	0.9606	0.0406	0.2015	0.0406

a) Precision, Recall, and F1-Score

Figure 2 compares the precision, recall, and F1-score of the three models. The CNN demonstrates the maximum scores across all three metrics, indicating its balanced performance in minimizing false positives and false negatives.

Received: 01/07/2025; Revised: 05/07/2025; Accepted: 05/07/2025; Published: 08/08/2025

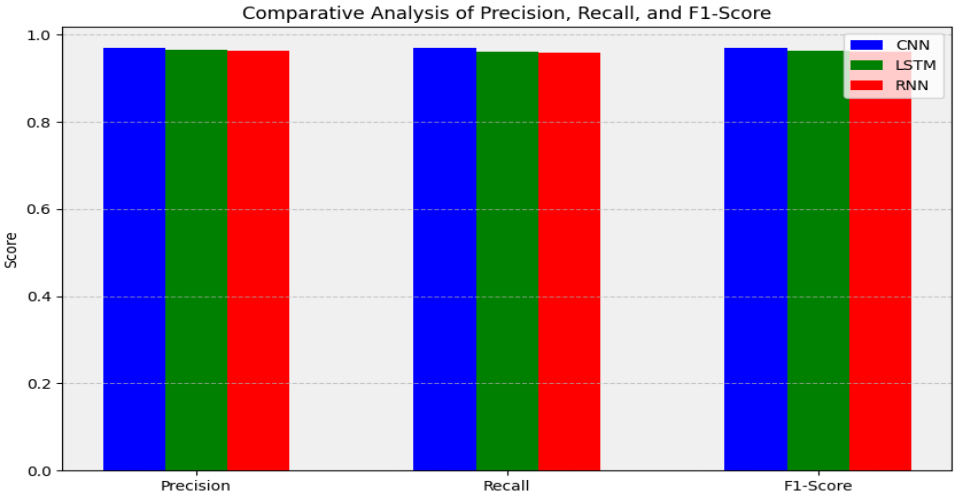

Figure 2: Comparative analysis of precision, recall, and F1-score

b) Error Metrics

Figure 3 illustrates the error metrics (MSE, RMSE, MAE) of the models. The CNN consistently shows the lowest values, indicating fewer prediction errors compared to the LSTM and RNN models.

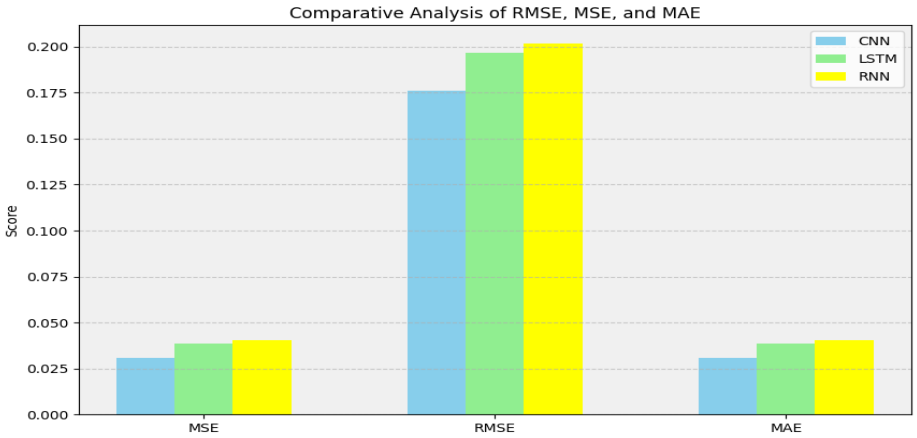

Figure 3: Comparative analysis of RMSE, MSE and MAE

c) Accuracy

Figure 4 depicts the accuracy comparison. The CNN attained the maximum accuracy (96.90%), subsequently the LSTM (96.13%) and the RNN (95.93%). This confirms the CNN's superior ability to classify messages correctly.

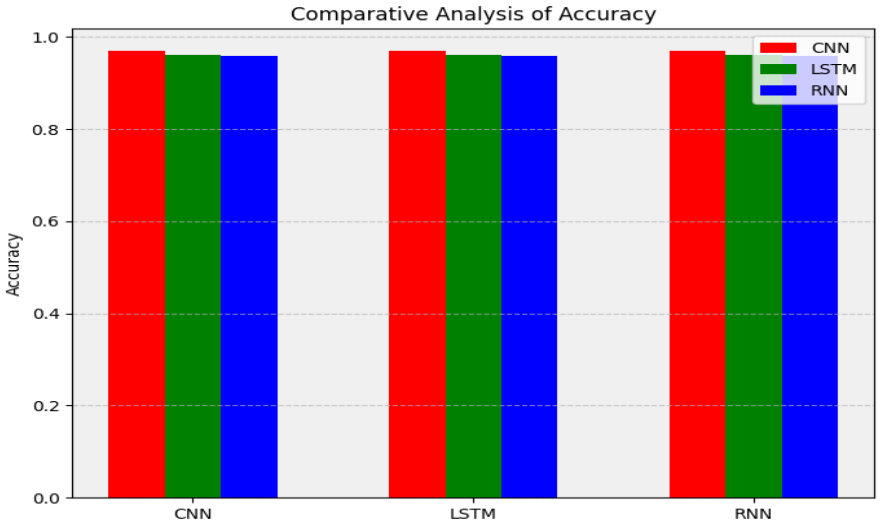

Figure 4: Comparative analysis of accuracy

Received: 01/07/2025; Revised: 05/07/2025; Accepted: 05/07/2025; Published: 08/08/2025

d) Confusion Matrix Analysis

Figure 5, 6, and 7 revealed the confusion matrices for the three deep learning models CNN, LSTM, and RNN used in SMS spam detection. Each confusion matrix illustrates the classification performance by displaying the number of correctly and incorrectly classified instances.

i. **CNN:** The CNN model demonstrated strong performance, with 883 correctly identified 'ham' messages (TN) and 133 accurately classified 'spam' messages (TP). The model misclassified 6 'ham' messages as 'spam' (FP) and 12 'spam' messages as 'ham' (FN). These results reflect the CNN's capability to minimize misclassification errors effectively.

ii. **LSTM:** The LSTM model identified 880 'ham' messages (TN) and 131 'spam' messages (TP) correctly. It misclassified 9 'ham' messages as 'spam' (FP) and 14 'spam' messages as 'ham' (FN). The slightly higher FN count compared to CNN indicates that LSTM has marginally lower sensitivity to detecting 'spam' messages.

iii. **RNN:** The RNN model achieved accurate classification for 883 'ham' messages (TN) and 128 'spam' messages (TP). However, it incorrectly labeled 6 'ham' messages as 'spam' (FP) and 17 'spam' messages as 'ham' (FN). The increased FN count suggests that RNN has a reduced capacity for capturing intricate spam patterns compared to CNN and LSTM.

iv.

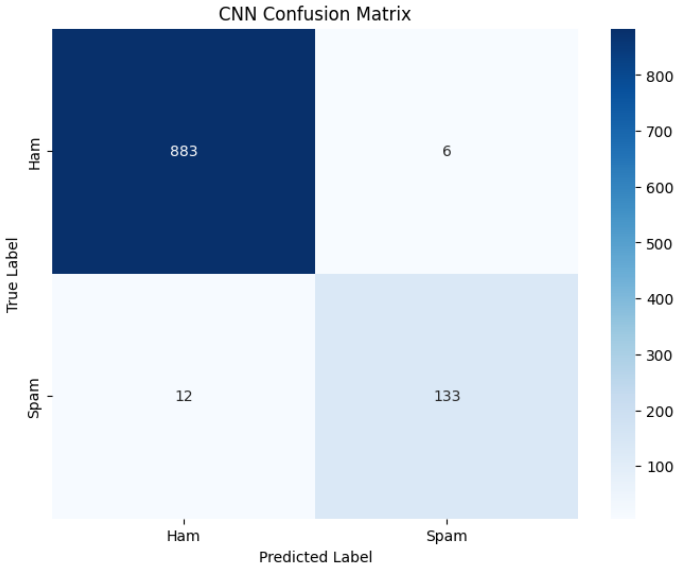

Figure 5: Confusion matrix for CNN model

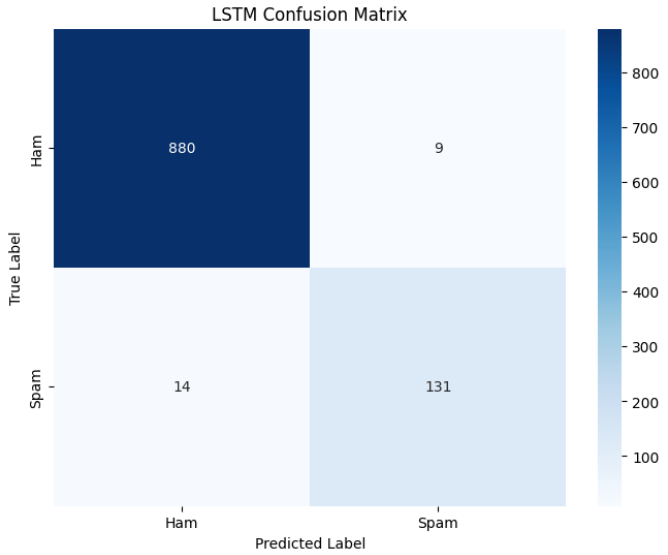

Figure 6: Confusion matrix for LSTM model

Received: 01/07/2025; Revised: 05/07/2025; Accepted: 05/07/2025; Published: 08/08/2025

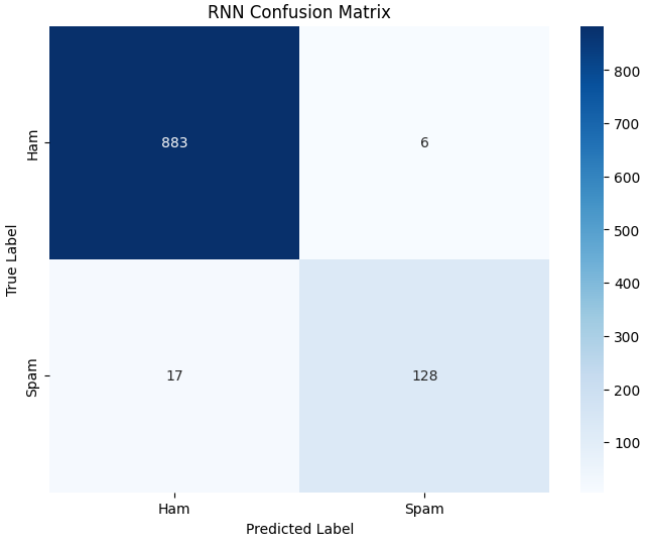

Figure 7: Confusion matrix for RNN model

The confusion matrices confirm that the CNN model provides superior performance in distinguishing between 'ham' and 'spam' messages, with fewer false positives and negatives than LSTM and RNN. This indicates that CNN is the most effective model for SMS spam detection among the evaluated algorithms. The results demonstrated that the CNN model superseded the LSTM and RNN models across all evaluation metrics. Its ability to extract local patterns from TF-IDF vectorizer input contributed to its success. While the LSTM performed better than the RNN, it was not as effective as the CNN in handling the specific characteristics of SMS data. Based on these findings, the CNN was selected as the most effective model for the SMS spam recognition system.

4.0 Conclusion

The rapid growth of mobile communication and widespread use of SMS services have resulted in an increase in unsolicited and malicious messages, commonly referred to as SMS spam. This study employed deep learning techniques to develop an operative SMS spam recognition system, evaluating three models: CNN, LSTM, and RNN. Among these, the CNN model established superior performance, attaining an accuracy of 96.90%, precision of 0.9692, recall of 0.9690, F1-score of 0.9691, and the lowest error metrics (MSE: 0.0309, RMSE: 0.1759, MAE: 0.0309). The CNN model's high recall and precision highlight its reliability in identifying both spam and legitimate messages, making it the most effective solution for SMS spam detection. Despite these achievements, the study has limitations, including the exclusive use of supervised deep learning techniques and the manual selection of the best-performing model without multi-criteria decision-making methods. Additionally, the study focused only on SMS spam detection and did not explore applications across other communication platforms, such as web-based applications or email filtering. Future research should explore hybrid deep learning models, unsupervised learning techniques, and deploy solutions across multiple communication platforms to enhance scalability and robustness. Subjecting the system to live testing with robust and more diverse datasets would also provide deeper insights into its global applicability. This study offers significant benefits to developers, institutions, and organizations, serving as groundwork for impending revolutions in spam detection and beneficial to the practical deployment of machine learning in communication security systems. It is recommended that, application of federated leaning should be employed for SMS spam modelling to further enhance model performance.

References

[1] A. Bhowmick and S. M. Hazarika, "E-mail spam filtering: A review of techniques and trends," in *Lecture Notes in Electrical Engineering*, 2018, pp. 583–590. doi: 10.1007/978-981-10-4765-7_61.

[2] O. Abayomi-Alli, S. Misra, and A. Abayomi-Alli, "A deep learning method for automatic SMS spam classification: Performance of learning algorithms on indigenous dataset," *Concurr. Comput. Pract. Exp.*, vol. 34, no. 17, 2022, doi: 10.1002/cpe.6989.

[3] H. Mandar, Shivaji, "SMS Spam Classification Using Machine Learning," *Lect. Notes Electr. Eng.*, vol. 967, pp. 41–47, 2023, doi: 10.1007/978-981-19-7169-3_4.

[4] F. B. Zamzuri, "Types and Methods of Managing SPAM Messages: A Review," 2023.

[5] S. Jane, M. Buckley, and D. Greene, "Expert Systems with Applications SMS spam filtering: Methods and data," *Expert Syst. Appl.*, vol. 39, no. 10, pp. 9899–9908, 2012, doi: 10.1016/j.eswa.2012.02.053.

Received: 01/07/2025; Revised: 05/07/2025; Accepted: 05/07/2025; Published: 08/08/2025

[6] I. Sharon, "SMS Spam Detection in Nigeria. Team Members: Praise, Habeebullah… | by Sharon Ibejih | Medium." Accessed: Mar. 22, 2024. [Online]. Available: https://sharonibejih.medium.com/sms-spam-detection-in-nigeria-76b149498bd7

[7] T. J. Rani, T. J. Vumesh, P. Saiteja, V. A. K. Reddy, and M. Meghana, "SMS Spam Detection Framework Using Machine Learning Algorithms and Neural Networks," *Int. J. Comput. Sci. Mob. Comput.*, vol. 10, no. 6, pp. 10–19, 2021, doi: 10.47760/ijcsmc.2021.v10i06.002.

[8] T. Avi Kumar, "The Fundamentals of Natural Language Processing: A Beginner's Guide | DataKwery." Accessed: Mar. 20, 2024. [Online]. Available: https://www.datakwery.com/post/fundamentals-of-nlp-guide/

[9] I. H. Sarker, "Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions," *SN Comput. Sci.*, pp. 1–20, 2021, doi: 10.1007/s42979-021-00815-1.

[10] A. A. Awoseyi *et al.*, "Enhancing car plate recognition with convolutional neural network and regular expressions correction," vol. 13, no. 2, pp. 2073–2080, 2024, doi: 10.11591/ijai.v13.i2.pp2073-2080.

[11] K. B. Pradeep, "SMS Spam Detection Using Machine Learning and Deep Learning Techniques," *J. Phys. Conf. Ser.*, vol. 1797, no. 1, pp. 1–55, 2021, doi: 10.1088/1742-6596/1797/1/012017.

[12] G. Paaß and S. Giesselbach, *Foundation Models for Natural Language Processing -- Pre-trained Language Models Integrating Media*. 2023. [Online]. Available: http://arxiv.org/abs/2302.08575

[13] O. Agboola, "Spam Detection Using Machine Learning and Deep Learning," 2022.

[14] H. Lee, S. Jeong, S. Cho, and E. Choi, "Visualization Technology and Deep-Learning for Multilingual Spam Message Detection," *Electron.*, vol. 12, no. 3, 2023, doi: 10.3390/electronics12030582.

[15] P. K. Roy, J. P. Singh, and S. Banerjee, "Deep learning to filter SMS Spam," *Futur. Gener. Comput. Syst.*, vol. 102, pp. 524–533, 2020, doi: 10.1016/j.future.2019.09.001.

[16] A. L. Gawade, S. S. Shinde, S. G. Sawant, R. S. Chougule, and A. A. Mahaldar, "A Review of SMS Spam Detection," vol. 868, no. 10, pp. 1088–1091, 2023.

[17] G. M. Shahariar, S. Biswas, F. Omar, F. M. Shah, and S. Binte Hassan, "Spam Review Detection Using Deep Learning," *2019 IEEE 10th Annu. Inf. Technol. Electron. Mob. Commun. Conf. IEMCON 2019*, pp. 27–33, 2019, doi: 10.1109/IEMCON.2019.8936148.

[18] I. AbdulNabi and Q. Yaseen, "Spam email detection using deep learning techniques," *Procedia Comput. Sci.*, vol. 184, no. 2019, pp. 853–858, 2021, doi: 10.1016/j.procs.2021.03.107.

[19] A. Ghourabi and M. Alohaly, "Enhancing Spam Message Classification and Detection Using Transformer-Based Embedding and Ensemble Learning," *Sensors*, vol. 23, no. 8, pp. 1–17, 2023, doi: 10.3390/s23083861.

[20] A. Ghourabi, M. A. Mahmood, and Q. M. Alzubi, "A hybrid CNN-LSTM model for SMS spam detection in arabic and english messages," *Futur. Internet*, vol. 12, no. 9, pp. 1–16, 2020, doi: 10.3390/FI12090156.

[21] S. Raga and M. Chaitra, "Machine Learning and Deep Learning Techniques for SMS Spam Detection, Accuracy Check and Comparative Study," *Int. J. Res. Publ. Rev.*, vol. 3, no. 5, pp. 2601–2604, 2022, [Online]. Available: www.ijrpr.com

[22] P. K. Roy, J. P. Singh, and S. Banerjee, "Deep learning to filter SMS Spam," *Futur. Gener. Comput. Syst.*, vol. 102, pp. 524–533, Jan. 2020, doi: 10.1016/J.FUTURE.2019.09.001.

[23] O. N. Akande *et al.*, "SMSPROTECT: An automatic smishing detection mobile application," *ICT Express*, vol. 9, no. 2, pp. 168–176, 2023, doi: 10.1016/j.icte.2022.05.009.

[24] M. Nivaashini, R.S.Soundariya, A.Kodieswari, and P.Thangaraj, "SMS spam detection using Deep Neural Network," *Int. J. Pure Appl. Math.*, vol. 119, no. 18, pp. 2425–2436, 2018.

[25] P. Nivaashini, M., Soundariya, R. S, Kodiieswari, A. & Thangaraj, "SMS Spam Detection Using Deep Neural Network," *Int. J. Pure Appl. Math.*, vol. 119, no. 18, pp. 2425–2436, 2018.

[26] F. E. Ayo, L. A. Ogundele, S. Olakunle, J. B. Awotunde, and F. A. Kasali, "A hybrid correlation-based deep learning model for email spam classification using fuzzy inference system," Decis. Anal. J., vol. 10, p. 100390, 2023, doi: 10.1016/j.dajour.2023.100390.

Received: 01/07/2025; Revised: 05/07/2025; Accepted: 05/07/2025; Published: 08/08/2025