IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received 26 January 2024, accepted 9 February 2024, date of publication 12 February 2024, date of current version 21 February 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3365193

APPLIED RESEARCH

Uncovering SMS Spam in Swahili Text Using Deep Learning Approaches

IDDI S. MAMBINA[1], **JEMA D. NDIBWILE**[2], **DEO UWIMPUHWE**[2], **AND KISANGIRI F. MICHAEL**[1]

[1]The Nelson Mandela African Institution of Science and Technology, Arusha 23306, Tanzania
[2]College of Engineering, Carnegie Mellon University Africa, Kigali, Rwanda

Corresponding author: Iddi S. Mambina (mambinai@nmaist.ac.tz)

This work was supported in part by the University of Dodoma.

ABSTRACT In today's communications, Short Message Service (SMS) and Internet protocol-based messaging systems are the most widely used channels. These services are currently the target of an unprecedented number of threats due to their rising appeal. Some spammers have more nefarious motives, even though most spam stems from businesses seeking to promote their products. In recent years, as new security threats such as Smishing have emerged, the amount of SMS spam has experienced substantial growth. The scientific community has paid less attention to SMS spam than email spam even though both are extensively utilized. SMS spam presents extra processing hurdles. These include the use of lexical variants, SMS-like contractions, or sophisticated obfuscations, which degrade the performance of conventional filtering solutions. In this study, we investigate the efficacy of deep-learning models for filtering Swahili SMS spam based on linguistics and behavioral patterns using a real-world dataset from telecommunications companies in Tanzania. To validate the effectiveness of our models we subjected them to the UCI dataset of spam messages written in English. The models were trained and tested with 10 k-fold cross-validation. The experimental results show that the CNN-LSTM-LSTM hybrid model attained the highest accuracy of 99.98 on the Swahili dataset while CNN-BiLSTM performed better on the UCI dataset with an accuracy of 98.38.

INDEX TERMS Deep learning, natural language processing, Swahili, SMS, spam detection.

I. INTRODUCTION

SMS has experienced little to no decline in popularity since Neil Papworth delivered the first text message in December 1992 [1]. To this day, SMS remains the most popular type of text messaging service. It enables individuals to communicate with one another meaningfully. Furthermore, businesses also believe that SMS marketing is the most effective way for companies to interact with consumers, especially for promotions that require instant impact. Despite the prevalence of messaging apps like Messenger, Telegram, WhatsApp, and WeChat on the market, classical SMS communication continues to be unmatched in its versatility. Moreover, the SMS application is preinstalled on all mobile devices, unlike other messaging applications, and there are no registration or sign-in requirements. There are 6.64 billion active SMS users worldwide, representing 83.96% of the world population who own a mobile phone [2]. With a 98% text opening rate, SMS demonstrates itself to be the most effective delivery method. The average person checks their phone 160 times per day. SMS can be used to promote, increase customer interaction, and update users on various activities [3]. The low cost of sending text messages, the rapid increase in mobile phone subscribers, the ease of use, and the fact that SMS does not require internet connectivity to be delivered all contribute positively to the popularity of SMS for spammers [3], [4].

Malicious activities have grown in tandem with the exponential growth of SMS sent across networks. Distributing spam via SMS is not only irritating to users but also costs both users and businesses a lot of money. According to Ghourabi et al., SMS spam affects 68% of mobile phone subscribers [5]. SMS spam can be defined as any irrelevant short message sent to a user via a mobile network [6]. Smishing occurs when SMS spam contains malicious content [5].

The associate editor coordinating the review of this manuscript and approving it for publication was Arianna Dulizia.

Smishing is a combination of two terms: "SMS and Phishing". It is a type of scam similar to email phishing that uses short message services. The objective of this attack is to send text messages to mobile users to steal personal information, obtain money, or coerce users to take unintended actions. Smishing has become a major annoyance for mobile subscribers as a result of its pervasiveness [3]. Themes that frequently appear in Smishing messages include lottery winning, credit card fraud, gift winning, and mobile money fraud. These messages typically ask users to click a link, dial a number, cancel a transaction, or send an email to an address provided.

Compared to email phishing, Smishing attack filtering is still lacking. The majority of previous research on Smishing attacks relied on the manual engineering of features before classification. Feature importance and information gain graphs are additionally employed to increase classification accuracy [6]. Deep-learning approaches have made an effort to scale back the use of manual feature engineering. Deep-learning model components learn on their own and self-train to allow unsupervised acquisition, avoiding the time-consuming phase of feature selection and extraction [3]. Many disciplines, including image classification [7], [8], speech recognition [9], intrusion detection [10], natural language processing [11], [12], [13], [14], and email spam classification [5] have shown promising results using deep-learning approaches. Researchers have not treated deep neural networks in much detail, though, when it comes to classifying Smishing messages. Additionally, sentiment classification [15], and language modeling [16], [17], [18], [19] are the only applications of deep-learning for languages with resource constraints, such as Swahili.

The world is home to 7,000+ languages, of which the vast majority remain understudied. Natural Language Processing (NLP) research has focused mostly on 20 languages out of the several thousand available [20]. The expression "low-resource languages" is frequently used to describe understudied languages. Additionally, low-resource languages are less computerized, less privileged, less commonly taught, or have a lower density [20], [21], [22], [23]. We borrow the definition of "low-resource languages" from [20], where "low-resource" indicates languages with less digital data, which hinders statistical data processing models from being applied. We categorize Swahili as one of the low-resource languages. Swahili is among the 10 most widely spoken languages in the world and the most widely spoken in Africa, with over 200 million speakers [24], [25]. Furthermore, Swahili is the native language of 60 million Tanzanians, of whom 140 million speak it as a second language in East, Central, and Southern Africa, which serves as an inspiration for this study. In addition, countries such as South Africa, Namibia, and Botswana are considering its introduction into schools [26], [27].

This study set out to explore SMS spam messages delivered in Swahili-speaking countries using a real-world Swahili SMS spam dataset. The contribution of this study, organized and carried out on Swahili spam text settings, is as follows:

- Introduction of a hybrid deep-learning model to effectively classify Swahili SMS spam text.
- Evaluating the performance of the proposed model by comparing it with other deep-learning models.
- We reviewed and categorized existing deep-learning approaches for SMS message detection in low-resourced languages.
- The proposed model achieved a remarkable accuracy of 99.98%.
- The effect of the number of iterations per epoch during training of the proposed model is analyzed.
- The impact of hyperparameter turning on obtaining the best-fit parameter of our proposed model is well highlighted.

In addition, the effect of using different deep-learning techniques is discussed in this paper by comparing the accuracies of several deep-learning architectures. Furthermore, the performance of the proposed model was evaluated on an English SMS spam dataset to set a baseline. The remainder of this paper is organized as follows: Section II presents related works; Section III describes materials and methods; Section IV presents results and discussion; and Section V summarizes the conclusion and future work.

II. RELATED WORKS

Recently, researchers proposed hybrid models to improve the detection of SMS phishing attacks. To filter SMS spam, Baaqeel & Zagrouba suggested a hybrid machine-learning model of SVM and K-Means [28]. To achieve the best results, the implementation was head-to-head, with an unlabeled dataset fed into the K-Means clustering model, and the results were fed into an SVM classifier. If both models agree, a label is classified as spam or ham. The proposed model achieved an accuracy of 98.8%. Although the model worked well for the proposed problem when applied to the Swahili dataset, it performed poorly, returning an accuracy of 94.8%. Ghourabi et al., investigated various deep-learning models for classifying text messages written in Arabic and English [5]. They tested twelve different models before settling on a combination of Convolution Neural Network (CNN) and Long Short-Term Memory (LSTM) to produce an improved neural network model (CNN-LSTM) that outperformed other models. They evaluated their model using accuracy, precision, recall, and area under receiver operator characteristics curve (ROC-AUC) metrics. The model achieved an accuracy of 98.3%, precision of 95.3%, recall of 91.4%, and ROC-AUC of 93.7%. In contrast to the achieved accuracy on the Arabic and English datasets, when a similar model was applied to the Swahili dataset, it returned an accuracy of 97%.

A deep-learning model by Roy et al., with 99.44% accuracy for filtering SMS spam by categorizing it as spam or non-spam [6]. They compared different classifiers using three-layered CNN and LSTM models. The three-layered CNN model, with dropout values ranging

TABLE 1. Analysis of the proposed approach with existing approaches.

LITERATURE	YEAR OF PUBLICATION	DATASET	TECHNIQUE EMPLOYED	PROPOSED ARCHITECTURE	RESEARCH AIM	PERFORMANCE (ACCURACY)
[3]	2020	English (UCI)	Deep learning	Random multimodel deep learning	To classify text messages written in English as spam or ham	99.26 %
[5]	2020	Arabic + English (UCI)	Deep learning	CNN-LSTM	To classify text messages as spam or ham written in Arabic and English	98.3%
[6]	2020	English (UCI)	Deep learning	3-Layered CNN	To classify text messages written in English as spam or ham	99.44%
[31]	2011	Hindi + English (UCI)	Machine learning	Bayesian model	To classify text messages written in Hindi and English	Not Applicable
[32]	2021	Bahasa Indonesia	Machine learning	Random forest	To classify text message written in Indonesian	94.62 %
Proposed	2024	Swahili	Deep Learning	CNN-LSTM-LSTM	Swahili Text Classification	99.98 %

from 0-1, performed the best. A study by Gomaa evaluated seven deep-learning models against six traditional machine-learning models [3]. The study utilized the University of California Irvine (UCI) classical spam dataset. In this dataset, the gradient-boosted tree model achieved an accuracy of 96.86%, whereas the Random Multimodal Deep Learning (RMDL) model achieved 99.26% accuracy. The study conducted by [29] focused on the application of the fastText model for the classification of emotional polarity in Chinese text. The choice of fastText was motivated by its capacity to generate concise, continuous, and high-quality representations. The model attained an accuracy of 92.86%. Simiraly, Kuyumcu et al., assessed the effectiveness of the fastText model using the TTC-3600 Turkish dataset [30]. The dataset underwent preprocessing via the Zemberek toolkit, and two feature selection techniques, namely correlation-based and attribute ranking-based methods, were employed to eliminate irrelevant features. The attained accuracy of this model was 93.52%. However, when the fastText model was applied to a Swahili dataset, the accuracy was only 60%, which is not particularly convincing, as it is nearly on par with random guessing.

Despite the push towards SMS spam classification, in most cases, researchers work with English datasets. However, few studies have focused on classifying SMS spam for low-resource languages. For instance, Ghourabi et al., proposed a model to detect SMS messages containing a mixture of Arabic and English text [5]. They collected a dataset of 2,730 text messages. However, they attested that their dataset was not significant for drawing a generalized conclusion. Yadav et al., proposed SMSAssanin to filter spam messages written in Hindi and English [31]. They used Bayesian models to filter SMS on a dataset containing 4,318 SMS for training. However, the plan was to collect more than 20,000 messages to obtain better representation. Theodorus et al., proposed a model to detect spam SMS in Bahasa Indonesia [32]. They gathered 4,125 SMS categorized as ham, spam, and promotional messages. Training was performed in two batches, and the results differed by a 12% margin between the first and second batches. They associated this difference with the dataset size. In general, the size of the dataset can be a limiting factor in the performance of the models. Compared with previous studies, our 32,000-message dataset stands out in terms of the number of messages gathered. Moreover, studies report that direct application of SMS filters results in low filtering accuracy due to the short length of SMS, the prevalence of shorthand abbreviations, and the size of labeled datasets that are available. The establishment of a benchmark dataset is further complicated by people's propensity to send SMS messages with different wordings based on their geographic location. Consequently, each region requires a unique dataset, and each model requires a distinctive set of hyperparameters to yield the best results.

This work builds on our previous study [33] and contributes to the limited body of literature on low-resource languages. The present study was designed to determine the effectiveness of employing deep learning models in categorizing Swahili SMS spam messages. In this study, owing to the shortage of the Swahili SMS spam dataset, we devised a data collection strategy and gathered over 32,000 Swahili text messages from the field. The findings should assist the general population in Swahili-speaking countries in making more informed decisions that would safeguard them from fraud and ensure their financial freedom.

A literature outline and comparative analysis of our proposed research study and existing research approaches are presented in Table 1. The analysis encompasses the literature, the year of publication for the particular literature, the dataset used for model building, the proposed approach, the analysis remarks, and the technique utilized.

III. MATERIAL AND METHODS

This section describes the approach and tools used to execute SMS spam detection tasks. In most cases, any NLP task involves data collection, data pre-processing, feature extraction, model training, and model evaluation. After data collection and pre-processing, the architecture introduces three parts to further process the collected texts. Word embedding and convolutional neural networks were used for weighting and feature selection, and a long short-term memory model was used for classification.

A. MODEL OVERVIEW

The main goal of this study is to use a deep-learning approach to classify Swahili SMS spam messages and distinguish them from ham messages. To train our dataset with two distinct labels, we chose five different deep-learning models: CNN, LSTM, Gated Recurrent Unit (GRU), Bidirectional LSTM (BiLSTM), and Bidirectional Encoder Representation from Transformers (BERT). These models were selected because they have shown promising results when applied to language modeling, as attested by the authors in [5], [13], [14], and [34]. These models are best known for sequence-modeling tasks. The use of a backpropagation algorithm to calculate the gradient that is specific to sequence data differentiates them from simple perceptron models or classical machine-learning models. Input data goes through a loop and passes to the middle layer of the network. Thereafter, the activation function, biases, and weights are standardized so that each layer has the same parameters [35]. The models were then trained based on the errors from the output layer to the input layers as a recurrent function.

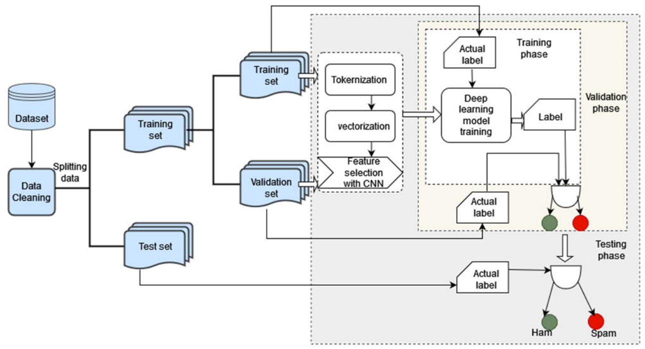

FIGURE 1. A methodological block diagram.

Figure 1 depicts generic steps that we took to classify Swahili text messages as spam and ham. The figure shows how data is preprocessed and fed into a deep-learning model for classification purposes. The deep-learning modules are further explained in the next subsection. Furthermore, the fastText model, which is an open-source model developed by Facebook Research as a tool for word vector and text classification, was used [29]. The fastText model can depict the model within a low-dimension, continuous space, allowing it to grasp the core meaning of the input text. fastText uses hierarchical softmax to predict the label of input text, which greatly reduces training time and increases accuracy. According to [36], fastText often matches the accuracy of a learning classifier while being significantly faster, both in terms of accuracy and evaluation. This out-of-the-box model was used to check if it could be used for the context at hand. The pseudo-code to implement our proposed method is given below.

```
# import necessary libraries
# load dataset and stopwords (data, stopwd)
# define function to preprocess text as preprocess
# define a function to resample the training data as resample
# define callbacks as callback
# define class MetricsCallback (validation_data, class names)
# define create_model
# define train_model
1:  Start
2:  load data and rename the column accordingly
3:  split the data into train and test set in a ratio of 80:20
4:  def preprocess(text)
5:     remove punctuation, stopwd, white spaces
6:     convert all text to small case
7:  end preprocess
9:  apply preprocess function to train and test data
10: map y-labels as ({'spam':1, 'leg':0})
11: def resample (x_train, y_train)
12:     vectorize ← initialize vectorizer
13:     transform (x_train)
14:     oversample ← initialize SMOTE
15: return oversample (x_train, y_train)
16: apply resample to train data
18: split resampled data with train_test_split in 80:20 ratio
19: use padding to create sequences of equal length
20: def callback
21:     es ← EarlyStopping()
22:     mc ← ModelCheckpoint()
23:     tb ← TensorBoard()
24:     red_lr ← LearningRateScheduler()
25: end callback
26: class MetricsCallback(inheriting keras callbacks)
27:     initialize class_names, validation_data
28:     def on_epoch(self, epoch, logs=none)
29:     val_features, y ← validation_data
30:     y_pred ← predict values from the model
31:     check evaluation metrics
32:     end on_epoch
33: end MetricsCallback
34: def create_model(maxlength, vocabsize)
35:     create input tensor
36:     pass it to embedding layer
37:     pass the results into a 1D convolution layer
38:     apply pooling layer
39:     pass the results into first LSTM layer
40:     pass the results into second LSTM layer
41:     pass the results into a dense layer
42:     define the model with inputs and outputs
43:     compile the model (optimizer, loss function, metric)
44:     start profiler and record the logs
45:     stop the profiler
```

```
46:     return the model
47: def train_model()
48:     split the data into 10 folds and iterate through them
49:     create a new model for each fold
50:     call MetricsCallback
51:     fit the model (data, epochs, and all callbacks)
52: end train_model
```

B. DEEP LEARNING MODELS

A deep neural network, as opposed to a simple perceptron network with one input, one output, and a hidden layer, has more than three layers, including input and output. This concept can be defined as an architecture with multiple hidden layers. Every layer in the deep-learning architecture is trained on a different set of features based on the execution of the previous layer. The deep layers of the network identify complex characteristics by blending information from the previous layers with more abstract information. The primary method for extracting data features in deep-learning is the convolution neural network [37]. The CNN consists of six distinct layers: input, convolution, activation, pooling, fully connected, and output layer. The basic mapping relationship of the CNN-LSTM model proposed is depicted in Figure 2. Where: $x^1 \ldots,\ x^n$ represents the characteristics map of the input words; $w^{i,1}$ and $w^{i,j}$ are the filters; b^i represents the bias; and y^i denotes the characteristics map of the output. The output of CNN is propagated to the LSTM cell for text classification.

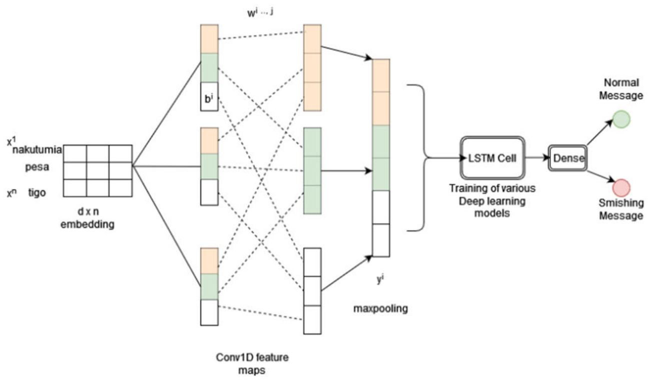

FIGURE 2. **A proposed deep-learning architecture.**

1) CONVOLUTION NEURAL NETWORK (CNN)

The convolution model that we present was adopted from the work presented in [37]. The CNN model mainly works by creating a word matrix at first, supposedly an input vector $x \in \mathbb{R}^{dxn}$. Furthermore, it defines a window vector that contains k consecutive words.

$$w_j = [x_j, x_{j+1}, \ldots, x_{j+k-1}] \tag{1}$$

It then identifies the hidden features of the text by applying a convolution operation to a window vector along with a filter $P \in \mathbb{R}^{kxn}$. Each feature element c_j is calculated as follows:

$$c_j = f(w_j \odot P + b) \tag{2}$$

where $\odot$ is an element-wise operator, b is a bias term and f is a nonlinear function. The nonlinear function used in our model is a rectified linear unit (ReLu) which is defined as:

$$f(x) = \max(0, x) \tag{3}$$

The nonlinear function helps to classify messages into predefined classes. A three-hundred-dimension embedding was used for the first channel, a two-hundred-dimension embedding for the second, and one-hundred-dimension for the final channel. The vocabulary's overall word count served as the input for each tier. With a kernel size of four, a pooling size of five, and a dropout of 30% for each iteration, we achieved our targets. After being flattened, the vector is then passed on for further processing.

2) LONG SHORT-TERM MEMORY (LSTM)

Long Short-Term Memory (LSTM) is a variant of Recurrent Neural Networks (RNN) that propagates historical information through a chain-like neural network. According to [38], with an appropriate design, LSTM can learn to bridge time gaps even with noisy, incompressible input sequences without sacrificing its short-time lag ability. As depicted in Figure 1, the LSTM architecture contains a range of repeated units for each time step. At every time-step, the network looks at the current input x_t as well as the previous output of the hidden state h_{t-1} and yields an output. The process passes through various gates, namely the input gate (i_t), memory cell (c_t), forget gate (f_t), hidden state (h_t) and output gate (o_t). The input vectors in the architecture have the same dimensions. The LSTM transition function definitions are according to [37]:

$$i_t = \sigma(w_i . [h_{t-1}, x_t] + b_i \tag{4}$$

$$f_t = \sigma(w_f . [h_{t-1}, x_t] + b_f \tag{5}$$

$$q_t = \tanh(w_q . [h_{t-1}, x_t] + b_q) \tag{6}$$

$$o_t = \sigma(w_o . [h_{t-1}, x_t] + b_o \tag{7}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot q_t \tag{8}$$

$$h_t = o_t \odot \tanh c_t \tag{9}$$

To decipher the notation used here, (x_t) is the input vector to the LSTM cell, σ is a sigmoid function that outputs [0, 1], tanh is a hyperbolic tangent function that controls what remains in the memory cell and it outputs values ranging from $[-1, 1]$, and the operator $\odot$ is an element-wise operator. Let's regard (f_t) as the function that controls the extent to which information from an old memory cell is going to be thrown away, (i_t) as the function that controls how much information is going to be stored in the current memory cell, (q_t) as the function that controls what remains in the memory cell, and (o_t) as the function that controls what to output based on the memory cell.

C. DATASET

The dataset, which was derived from the research project in [31], consists of 31,962 legitimate messages gathered over

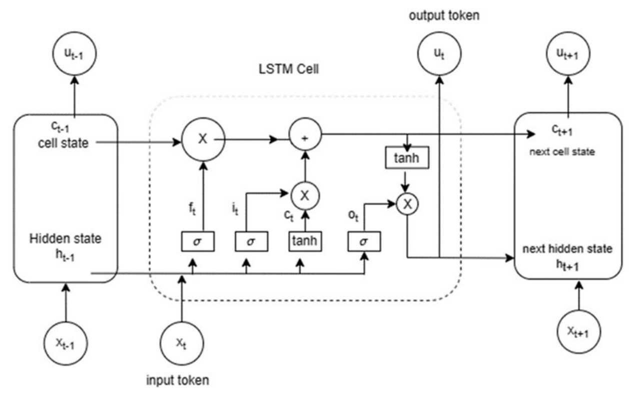

FIGURE 3. Basic structure of LSTM model.

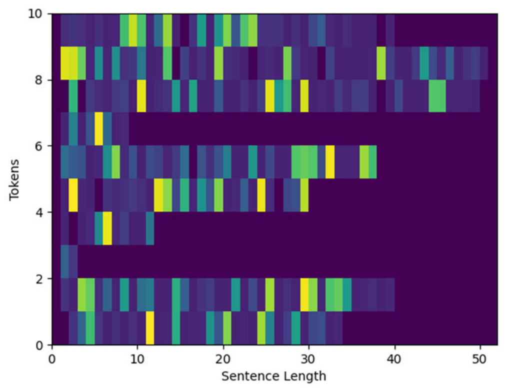

FIGURE 4. Colored block of tokens.

five months, from February to June 2021. Only 297 unique Smishing messages were retrieved from mobile network operators out of the ten million two hundred and fifty-one thousand two hundred and ten (10,251,210) spam messages. This dataset was collected from mobile network operators in Tanzania, and a series of experiments were performed. Mobile network operators are purposefully selected based on their mobile money market share. To reduce bias, 20% of the data were held as a test set, and the remaining 80% were divided into training and validation sets at a ratio of 80:20. The dataset is highly imbalanced; therefore, the synthetic minority oversampling technique (SMOTE) was applied to the training and validation set to increase the sensitivity of the classifier. Upon applying SMOTE, the dataset increased to a total of sixty-three thousand nine hundred and eighteen samples (63918).

D. TEXT PREPROCESSING

A team of experts encoded the dataset manually and consistently. Python library functions were used for text pre-processing and data cleaning. The data were converted to lowercase, and punctuation marks were removed. Numeric values were not deleted, as they are an important aspect of fraudulent messages. A list of stop-words from Masua and Masasi [39] was used. Texts were tokenized using a built-in Keras tokenization function. Figure 4 shows a colored block of tokens, where the background color represents padding tokens. Colored blocks are our input_ids which represent input tokens to the model. Input_ids vary in length; hence, padding is applied to create input tokens of equal length. Padding ensures that we can process a batch of sentences together in parallel since a deep-learning model requires an input tensor with consistent dimensions. The study used post padding, taking the maximum length of 60 words per sentence with a vocabulary size of 44,200.

E. EMBEDDING LAYER

The embedding layer provides a dense representation of words and their relative meanings. These representations are in vector form, where a vector represents the projection of a word into a continuous vector space. A representation of our dataset was created based on text messages. Every message is considered a sequence of tokens: T_1, T_2, T_3, $\ldots T_n$ from a given corpus. Unique words from the corpus were grouped to create a vocabulary. Each token in the vocabulary is assigned a unique integer value. These integer values are randomly assigned first and are updated when the vector is passed on to the convolution layer. The output dimensions for our embedding ranged from 300 to 100 on different channels. There are off-the-shelf pre-trained word embedding models, such as word2vec [40] and GloVe [41]. However, we decided to train our embeddings because text messages are unique and do not share the same characteristics as the normal text documents used to train these pre-trained embeddings. We used an embedding layer followed by a convolution layer on every neural network that was experimented on for better feature selection. This study proposes a deep neural network to classify Swahili Smishing messages by combining CNN and two long short-term memory (LSTM) techniques to improve classification accuracy and reduce false positives and false negatives.

F. PROPOSED METHOD

Traditional machine-learning models use manually constructed features to classify texts as Smishing or normal messages. Features such as the number of words in a sentence, the presence of a phone number in the message, words with missing characters, and buzzwords are often used. The performance of these models depends significantly on these features. Deep-learning approaches remove this dependence by automatically extracting features from text messages and using them for model training. The proposed model comprises a configuration consisting of one CNN layer and two LSTM layers. The integration of a convolution neural network layer before an LSTM layer has proven to enhance the model capacity. While CNN is traditionally associated with image processing, they have demonstrated effectiveness in text classification tasks as well. This effectiveness stems from CNN's ability to adeptly capture local patterns and features in text, enabling them to learn to identify crucial n-grams. Additionally, CNN utilizes parameter sharing and local connectivity to learn hierarchical features, facilitating the extraction of important information from various text segments without sole reliance on global context. The inclusion of a max pooling layer within the convolution neural network aids in reducing the spatial dimensions of the data. Furthermore, the pooling layer's ability to reduce sequence length acts as a valuable asset for computational efficiency. Incorporating two LSTM layers following the CNN layer offers numerous advantages over a single LSTM layer. Staking more than one LSTM layer enables the model to establish hierarchical representation, with the initial layer capturing lower-level features and the subsequent layer handling more intricate and abstract patterns by processing the output of the preceding layer. In addition, stacked LSTM layers enhance the architecture's ability to capture long-range dependencies within sequences, thereby improving its comprehension of the input data. Moreover, the presence of two LSTM layers offers flexibility to experiment with different architectures and hyperparameters, facilitating the identification of the optimal configuration for the specific task at hand. Lastly, stacked LSTM layers often lead to improved accuracy and enhanced generalization capabilities.

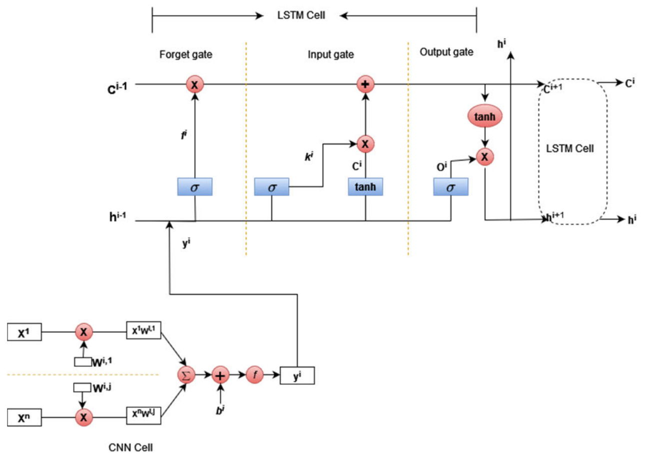

FIGURE 5. A CNN-LSTM-LSTM block diagram.

G. EXPERIMENT ENVIRONMENT

The experiments were carried out on a Linux online workstation preinstalled with Ubuntu 18.04.6 LTS (Long-Term Support) (Bionic Beaver), which was outfitted with an Intel Xeon (R) CPU @ 2.20 GHz and one NVIDIA Tesla T4 Graphical Processing Unit (GPU). The station was hosted by Google and accessed through Google Collaboratory Notebook. We used three channels, one for feature selection with the CNN model and two for classification with the LSTM architecture. We set each model to train for 20 epochs and applied an early stopping variable that looked at improving loss; the iteration terminated if the loss did not improve for three consecutive iterations. The training took approximately 20 minutes on average. TensorFlow served as the backend for the Keras deep-learning library. It took one hour and thirty minutes to complete training on all five models. The proposed architecture is illustrated in Figure 5.

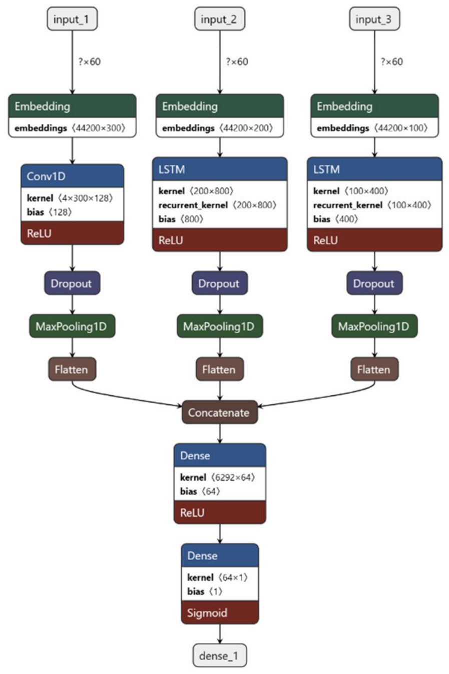

FIGURE 6. Summary of the proposed architecture.

H. PERFORMANCE METRICS

Choosing the appropriate evaluation metric is essential when assessing the performance of deep-learning models. Various metrics have been suggested for appraising deep-learning models in various contexts. In certain scenarios, particularly when dealing with imbalanced data, relying solely on a single metric may not offer a comprehensive understanding of the problem at hand. Consequently, it becomes necessary to utilize a combination of metrics to obtain a comprehensive assessment of the models. In light of this, we employed established metrics like accuracy, precision, recall, and the F1-score for performance assessment. However, before delving into the details of these metrics, it is important to define four key terms:

- **True Positive (TP):** The instance where the model accurately predicts a positive outcome and the actual result is indeed positive.
- **True Negative (TN):** The instance where the model accurately predicts a negative outcome and the actual result is indeed negative.
- **False Positive (FP):** The instance where the model predicts a positive outcome, but the actual outcome is negative.
- **False Negative (FN):** The instance where the model predicts a negative outcome, but the actual outcome is positive.

Accuracy: It serves as an effective gauge of the model's performance when there is an even distribution of classes. This metric is determined by dividing the total count of accurate predictions by the overall number of predictions.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{10}$$

Precision: When dealing with an imbalanced class distribution, it serves as a valuable measure of model performance. Precision is computed by dividing the total count of correctly identified positive instances by the total count of instances classified correctly. In essence, precision assesses the accuracy of the model in classifying instances as positive.

$$Precision = \frac{TP}{TP + FP} \tag{11}$$

Recall: It can be described as the proportion of positive instances that were accurately classified compared to the total count of positive instances. In essence, recall evaluates the capacity of the model to identify positive instances. A higher recall indicates that more positive examples are correctly identified.

$$Recall = \frac{TP}{TP + TN} \tag{12}$$

F1-Score: It is used to assess the algorithm's accuracy on the dataset, and it is well-suited for binary classification, which is the problem addressed in this study. It integrates both recall and precision and as such, it is defined as the harmonic mean of the algorithm precision and recall.

$$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{13}$$

IV. RESULTS AND DISCUSSION

The comprehensive results of the models will be discussed in this section. The experiments were conducted on a Swahili dataset gathered by [33] and a UCI dataset hosted online by Kaggle. There were 31,921 legitimate messages and 297 spam messages in the Swahili dataset, whereas the UCI dataset contained 4,825 ham messages and 747 spam messages. To assess the proposed approaches, we used false positive, false negative, f1-score, recall, precision, and accuracy. A confusion matrix is presented in Table 2, which shows how messages are placed as either spam or ham messages by the classifiers. It further helps to calculate the percentage of spam messages against legitimate messages.

The confusion matrix of the proposed model is shown in Figure 7. In this matrix, the model's predictions on a validation set are compared against the actual outcome. From the confusion matrix components, various metrics such as accuracy, precision and recall can further be delivered, providing a comprehensive understanding of the model's strengths and weaknesses.

The purpose of this study is to determine whether Swahili text messages contain social engineering content. With varying degrees of accuracy, all the selected models were able to distinguish Smishing from legitimate messages. In this section, we provide experimental evaluations. Because the dataset was highly imbalanced, we initially tested the ability of our model to recall the messages separately. Table 3 lists the performance of these models in recalling legitimate messages.

Furthermore, Table 4 lists the performance of these models to recall Smishing messages.

TABLE 2. Confusion matrix.

	Predicted as Ham	Predicted as Spam
Labeled as Ham	True Positive (TP)	False Positive (FP)
Labeled as Spam	False Negative (FN)	True Negative (TN)

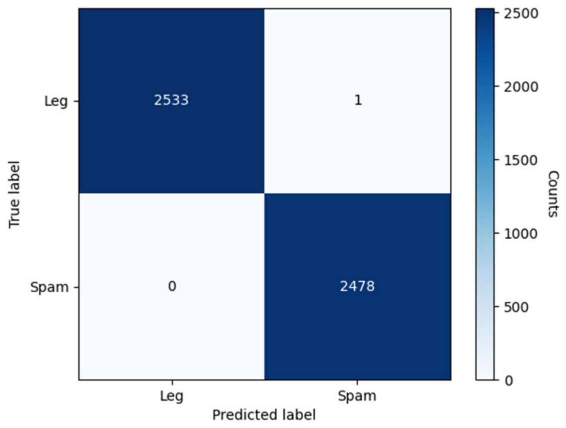

FIGURE 7. Confusion matrix of CNN-LSTM-LSTM model.

TABLE 3. Model ability to recall legitimate swahili messages.

Proposed Model	Accuracy	False Positive	False Negative	F1-Score
CNN-LSTM	99.99	1	0	100
CNN-LSTM-LSTM	99.99	1	0	100
CNN-GRU	99.99	1	0	100
CNN-BiLSTM	99.99	1	0	100

It is important to keep in mind the bias in previous results due to the exclusion of one class. However, with the small sample size of the Smishing message, caution must be exercised when training the models on a general dataset. Table 5 lists the results of the proposed approaches on the Swahili dataset. The CNN-LSTM-LSTM model outperformed the other models in terms of both f1-socre, accuracy, and number of false positives and false negatives, achieving an accuracy of 99.98%.

Table 6 shows that these models performed well on the validation datasets, whereas CNN-BiLSTM performed well compared to the other models. The results demonstrate a near-state-of-the-art performance for classifying SMS spam messages written in English. The results obtained are not far from the results presented by Goma et al., [3]. However, the substantial number of false positives indicates that the impact of user and language differences on model performance may be significant.

We conducted an experiment using the fastText model to investigate whether a simple neural network could achieve the desired outcome. The model was a shallow fastText with one input layer, one hidden layer and one output layer. It was trained with a 100-dimension word vectors learning rate set of {0.5, 1, 1.5, 2} over 50 epochs. Our findings presented in Table 7 indicate that augmenting the number of n-grams improves the performance of the task. However, the overall

TABLE 4. Model ability to recall smishing swahili messages.

Proposed Model	Accuracy	False Positive	False Negative	F1-Score
CNN-LSTM	100	0	0	100
CNN-LSTM-LSTM	100	0	0	100
CNN-GRU	100	0	0	100
CNN-BiLSTM	100	0	0	100

TABLE 5. Model performance evaluation on swahili dataset.

Proposed Model	Accuracy	False Positive	False Negative	F1-Score
CNN-LSTM	99.89	0	23	99.96
CNN-LSTM-LSTM	99.98	0	1	99.98
CNN-GRU	99.97	0	7	99.97
BERT	99.97	0	6	99.96
CNN-BiLSTM	99.91	0	13	99.90

accuracy, precision, and recall metrics do not show significant improvements. This suggests that classifying languages with morphological complexity and limited resources, such as Swahili, is challenging for a basic neural network. The model architecture and hyperparameters are not suitable for the problem, and we need a more complex model or a different neural network architecture. In Table 7, we present the result of fine-tuning the fastText model with different parameters in an attempt to enhance its accuracy, precision, and recall. The learning rate below 0.5 results in an accuracy that is below fifty percent, which implies the model is performing worse than random chances as it is classifying labels incorrectly more than getting them right. However, a higher learning rate often results in very unstable training, divergence, and failure to converge to an optimal solution. Hence, the fastText model was deemed unsuitable for classifying Swahili text.

Analysis of Experimental Results:

During the training process, we tuned three parameters namely: dropout, learning rate, and embedding dimension to increase the efficacy of our model. Finding appropriate values for these parameters would enable the model to converge better and avoid overfitting while improving model performance.

A. DROPOUT

During training, we select several dropout values, with the values taken into consideration being [0.1, 0.2, 0.25, and 0.3] accordingly. The optimal dropout value was selected based on the performance of the model. Figure 6 depicts the experimental results, which reveal that a dropout value of 0.2 suits most of the models except CNN, which performs best with a dropout value of 0.3. As a result, we used a dropout rate of 0.3 for the CNN model's feature selection stage before lowering it to 0.2 for the training of other models.

B. LEARNING RATE

As we update model parameters using gradient backpropagation, experimental evidence for learning rate and its impact on model performance is highlighted. The optimization algorithm we employed in this experimental process is called Adam, and it has a high computation efficiency and

TABLE 6. Model performance evaluation on UCI dataset.

Proposed Model	Accuracy	False Positive	False Negative	F1-Score
CNN-LSTM	97.80	46	3	91.17
CNN-LSTM-LSTM	97.49	55	1	89.70
CNN-GRU	97.71	48	3	90.77
BERT	97.93	22	2	95.63
CNN-BiLSTM	98.38	26	10	93.18

a fast convergence time. To improve the algorithm efficiency different learning rate values were selected for the experiments. Figure 7 shows the remarkable increase in accuracy for all models when the learning rate value is set to 0.0001. Since it produces the best results when compared to other learning rate values, 0.0001 was chosen as the learning rate for this study.

C. EMBEDDING DIMENSION

In the final part of hyperparameter tuning, we considered an embedding dimension. The number of features that the model can learn grows as the dimension increases. When dimensions considered are too small or excessive, this may easily result in an underfitting. This demonstrates the importance of choosing an appropriate embedding size for the model to train well. The outcome of the experimental analysis in which the model was subjected to various embedding dimensions is shown in Figure 8. The figure clearly shows that the models performed effectively with embedding dimensions between 100 and 200. This study employed an embedding dimension of 100 on the output channel after testing.

D. TRAINING AND VALIDATION

Figure 11 illustrates the variation in accuracy and loss over successful epochs during the training and validation phases. The training graph displays how well the model is learning from the training data, showing the convergence of the model prediction compared to the actual values. On the other hand, the validation curve depicts the model performance on a separate dataset that it has not seen during training, serving as a proxy for its ability to generalize to new, unseen data. There is a minimal discrepancy between the training and validation graphs, which provides a crucial indication that our model neither overfits nor underfits the data.

E. COMPUTATION COMPLEXITY OF THE MODEL

We measured the computation complexity of our model by considering the amount of computation resources, such as time and memory, required for training and inferences. We used cProfiler to retrieve the number of primitive calls and the total number of calls used to complete either a forward or backward pass of a neuron. Figure 12 compares the number of times a function is called at the lowest level, excluding calls made to other functions (primitive calls), against total calls, including both direct calls to a function and the number of calls made to other functions within that function. The result gives a more comprehensive view of the overall impact of

TABLE 7. fastText model results in classifying swahili text with fine-tuning parameters.

	n-gram =3				n-gram = 4				n-gram = 5			
	Learning rate (0.5)	Learning rate (1)	Learning rate (1.5)	Learning rate (2)	Learning rate (0.5)	Learning rate (1)	Learning rate (1.5)	Learning rate (2)	Learning rate (0.5)	Learning rate (1)	Learning rate (1.5)	Learning rate (2)
Accuracy	49	50	50	50	50	50	51	51	48	60	51	51
Precision	47	51	62	64	56	56	63	65	43	51	64	59
Recall	50	50	51	51	51	51	51	51	48	51	51	51
F1-Score	36	36	37	36	36	36	37	36	37	50	47	45

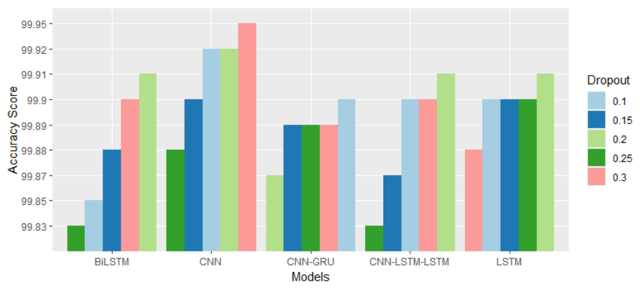

FIGURE 8. Summary of the dropout effect on model accuracy.

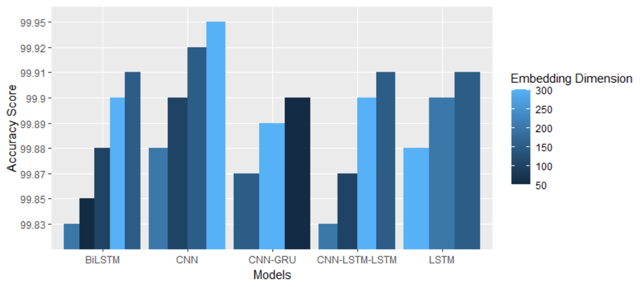

FIGURE 10. Summary of embedding effect on model accuracy.

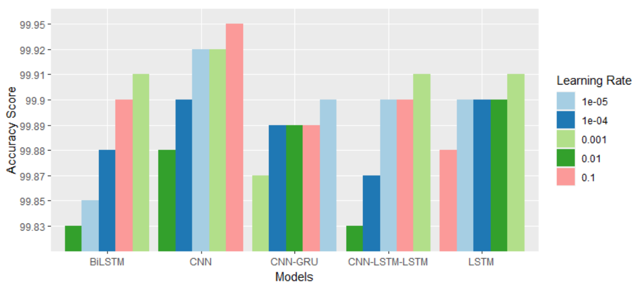

FIGURE 9. Summary of learning rate effect on model accuracy.

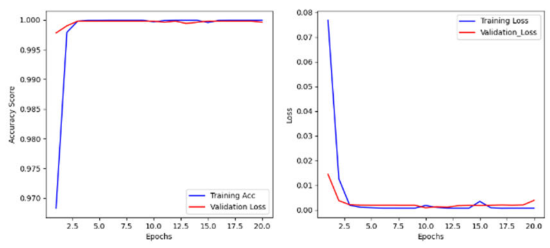

FIGURE 11. Training and validation graph.

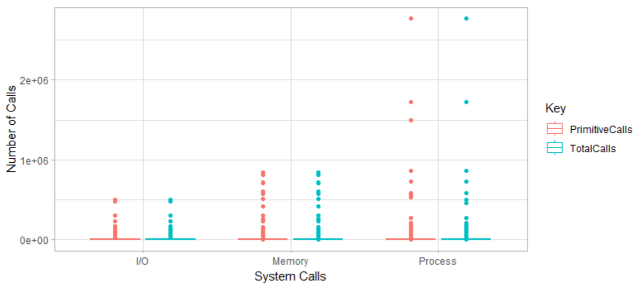

FIGURE 12. Number of calls made by the system.

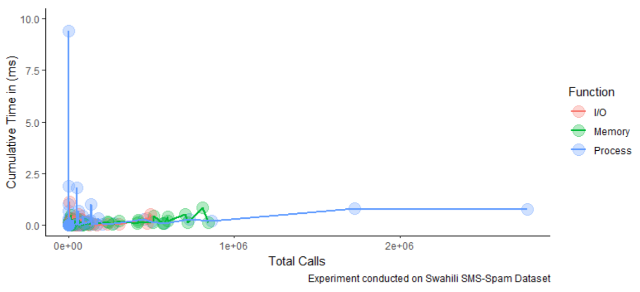

FIGURE 13. Comparing cumulative time and total calls.

the function on the program, including the calls it triggers internally.

The study further examined the total time spent in a particular function, including calls made to other functions within the profiled function. The results obtained from CProfiler analysis, cumulative time was used as a critical metric, offering valuable insight into the overall execution characteristics of the profiled function. Functions with low cumulative time, as depicted in Figure 13, are indicative that our model is less complex, while those with high cumulative time show potential performance bottlenecks, warranting closer scrutiny for optimization opportunities. The results indicate that most of our calls have a low cumulative time; hence, our model complexity is low.

1) COMPARISON AGAINST OTHER MODELS IN THE LITERATURE

Table 8 presents a comparison of the accuracy of our proposed model with other models that have undergone evaluation using datasets that are similar to Swahili in terms of language morphological structure. Languages such as Arabic, Turkish, and Bahasa Indonesia.

For these datasets, the best accuracy among existing research works using CNN-LSTM is 98.3% [5]. Our proposed model of CNN-LSTM-LSTM shows an accuracy of 99.98%. Furthermore, the proposed model performs better than other models implemented using fastText, RNN, Random Forest, CNN, and Random Multimodel Deep-Learning on spam text classification, with a margin of 0.6% from the

TABLE 8. Comparative accuracy against other models.

Model	Dataset	Accuracy
fastText [29]	Chinese data from Baidu Dianshi	92.86%
fastText [30]	TTC-3600 Turkish	93.52%
CNN-LSTM [42]	Expedia	98.25%
RNN [43]	Kaggle	87%
LSTM Autoencoder [44]	Dataset from Tubespam [45]	98%
CNN-LSTM [5]	Arabic + English	98.3%
3-Layered CNN [6]	UCI	99.44%
Random Forest [32]	Bahasa Indonesia	94.62%
CNN-LSTM-LSTM	Swahili	99.98%

best-performed model. The results demonstrate that our proposed model consistently performs better than other proposed models in the literature.

V. CONCLUSION AND FUTURE WORK

This study suggests using deep-learning approaches to distinguish between legitimate and fraudulent messages in Swahili text. The five deep-learning classifiers examined were CNN, LSTM, GRU, BiLSTM, and BERT. For model testing and training, 32,259 Swahili text messages were used as a dataset. We applied our models to the UCI dataset of English messages to further validate the robustness of our models. On the Swahili dataset, we unequivocally show that the CNN-LSTM-LSTM model outperforms other deep-learning models, and on the UCI dataset, the CNN-BiLSTM model outperforms other models. However, the model's ability to tune more hyperparameters could help it produce fewer false negatives. We plan to transform this model into a mobile application and release it for community use soon.

REFERENCES

[1] A. Ghourabi, "SM-Detector: A security model based on BERT to detect SMiShing messages in mobile environments," *Concurrency Comput., Pract. Exper.*, vol. 33, no. 24, pp. 44–59, 2021, doi: 10.1002/cpe.6452.

[2] P. Kaushik. *Daily SMS Mobile Usage Statistics 2021 | SMSEagle*. Accessed: Jun. 11, 2023. [Online]. Available: https://www.smseagle.eu/2022/01/26/daily-sms-mobile-usage-statistics-2021/

[3] W. H. Gomaa, "The impact of deep learning techniques on SMS spam filtering," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 1, pp. 544–549, 2020.

[4] O. N. Akande, O. Gbenle, O. C. Abikoye, R. G. Jimoh, H. B. Akande, A. O. Balogun, and A. Fatokun, "SMSPROTECT: An automatic smishing detection mobile application," *ICT Exp.*, vol. 9, no. 2, pp. 168–176, Apr. 2023, doi: 10.1016/j.icte.2022.05.009.

[5] A. Ghourabi, M. A. Mahmood, and Q. M. Alzubi, "A hybrid CNN-LSTM model for SMS spam detection in Arabic and English messages," *Future Internet*, vol. 12, no. 9, p. 156, Sep. 2020.

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

[7] S. Patel, R. Patel, N. Ganatra, and A. Patel, "Spatial feature fusion for biomedical image classification based on ensemble deep CNN and transfer learning," *Int. J. Adv. Comput. Sci. Appl.*, vol. 13, no. 5, pp. 1–7, 2022, doi: 10.14569/IJACSA.2022.0130519.

[8] A. Barodi, A. Bajit, A. Zemmouri, M. Benbrahim, and A. Tamtaoui, "Improved deep learning performance for real-time traffic sign detection and recognition applicable to intelligent transportation systems," *Int. J. Adv. Comput. Sci. Appl.*, vol. 13, no. 5, pp. 1–12, 2022, doi: 10.14569/IJACSA.2022.0130582.

[9] H. Isyanto, A. S. Arifin, and M. Suryanegara, "Voice biometrics for Indonesian language users using algorithm of deep learning CNN residual and hybrid of DWT-MFCC extraction features," *Int. J. Adv. Comput. Sci. Appl.*, vol. 13, no. 5, pp. 622–634, 2022, doi: 10.14569/IJACSA.2022.0130574.

[10] A. Alshehri, "Relational deep learning detection with multi-sequence representation for insider threats," *Int. J. Adv. Comput. Sci. Appl.*, vol. 13, no. 5, pp. 1–8, 2022, doi: 10.14569/IJACSA.2022.0130587.

[11] A. Berrajaa, "Natural language processing for the analysis sentiment using a LSTM model," *Int. J. Adv. Comput. Sci. Appl.*, vol. 13, no. 5, pp. 777–785, 2022, doi: 10.14569/IJACSA.2022.0130589.

[12] G. Abandah, A. Suyyagh, and M. Z. Khedher, "Correcting Arabic soft spelling mistakes using BiLSTM-based machine learning," *Int. J. Adv. Comput. Sci. Appl.*, vol. 13, no. 5, pp. 815–829, 2022, doi: 10.14569/IJACSA.2022.0130594.

[13] F. Alwajih, E. Badr, and S. Abdou, "Transformer-based models for Arabic online handwriting recognition," *Int. J. Adv. Comput. Sci. Appl.*, vol. 13, no. 5, pp. 898–905, 2022, doi: 10.14569/IJACSA.2022.01305102.

[14] M. J. Althobaiti, "BERT-based approach to Arabic hate speech and offensive language detection in Twitter: Exploiting emojis and sentiment analysis," *Int. J. Adv. Comput. Sci. Appl.*, vol. 13, no. 5, pp. 972–980, 2022, doi: 10.14569/IJACSA.2022.01305109.

[15] G. L. Martin, M. E. Mswahili, and Y.-S. Jeong, "Sentiment classification in swahili language using multilingual BERT," 2021, *arXiv:2104.09006*.

[16] N. D. Zacharia, M. K. Dahouda, and I. Joe, "Language-based classification of words using deep learning," in *Proc. Korea Inf. Process. Soc. Conf.*, 2021, pp. 411–414.

[17] C. S. Shikali and R. Mokhosi, "Enhancing African low-resource languages: Swahili data for language modelling," *Data Brief*, vol. 31, Aug. 2020, Art. no. 105951.

[18] C. S. Shivachi, R. Mokhosi, Z. Shijie, and L. Qihe, "Learning syllables using Conv-LSTM model for swahili word representation and part-of-speech tagging," *ACM Trans. Asian Low-Resource Lang. Inf. Process.*, vol. 20, no. 4, p. 58, 2021.

[19] C. S. Shikali, Z. Sijie, L. Qihe, and R. Mokhosi, "Better word representation vectors using syllabic alphabet: A case study of swahili," *Appl. Sci.*, vol. 9, no. 18, p. 3648, Sep. 2019.

[20] A. Magueresse, V. Carles, and E. Heetderks, "Low-resource languages: A review of past work and future challenges," 2020, *arXiv:2006.07264*.

[21] Y. Tsvetkov, "Opportunities and challenges in working with low-resource languages slides Part-1," Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep., 2017.

[22] A. K. Singh, "Natural language processing for less privileged languages: Where do we come from? Where are we going?" in *Proc. IJCNLP Workshop NLP Less Privileged Lang.*, 2008, pp. 1–6.

[23] C. Cieri, M. Maxwell, S. Strassel, and J. Tracey, "Selection criteria for low resource language programs," in *Proc. 10th Int. Conf. Lang. Resour. Eval. (LREC)*, 2016, pp. 4543–4549.

[24] B. E. Coleman, "A history of Swahili," *Black Scholar*, vol. 2, no. 6, pp. 13–25, 1971.

[25] UNESCO, "World Kiswahili language day," (2021). *41st Session, Paris: United Nations Education, Scientific and Cultural Organization*. Accessed: Jan. 29, 2022. [Online]. Available: https://unesdoc.unesco.org/ark:/48223/pf0000379702

[26] C. Sabao and O. Nauyoma, "On the 'proposed' introduction of Kiswahili to the Namibian school curriculum: Emerging perspectives," *J. Namibian Stud., History Politics Culture*, vol. 28, pp. 103–113, Dec. 2020.

[27] R. Wildsmith-Cromarty and A. N. Conduah, "Issues of identity and African unity surrounding the introduction of an exogenous African language, Swahili, at tertiary level in South Africa," *Int. J. Bilingual Educ. Bilingualism*, vol. 17, no. 6, pp. 638–653, Nov. 2014, doi: 10.1080/13670050.2014.953772.

[28] H. Baaqeel and R. Zagrouba, "Hybrid SMS spam filtering system using machine learning techniques," in *Proc. 21st Int. Arab Conf. Inf. Technol. (ACIT)*, Nov. 2020, pp. 1–8.

[29] T. Yao, Z. Zhai, and B. Gao, "Text classification model based on fastText," in *Proc. IEEE Int. Conf. Artif. Intell. Inf. Syst. (ICAIIS)*. China: IEEE, Mar. 2020, pp. 154–157, doi: 10.1109/ICAIIS49377.2020.9194939.

[30] B. Kuyumcu, C. Aksakalli, and S. Delil, "An automated new approach in fast text classification (fastText) A case study for Turkish text classification without pre-processing," in *Proc. 3rd Int. Conf. Natural Lang. Process. Inf. Retr.*, vol. 2019, pp. 1–4.

[31] K. Yadav, P. Kumaraguru, A. Goyal, A. Gupta, and V. Naik, "SMSAssassin: Crowdsourcing driven mobile-based system for SMS spam filtering," in *Proc. 12th Workshop Mobile Comput. Syst. Appl.*, New York, NY, USA, Mar. 2011, pp. 1–6, doi: 10.1145/2184489.2184491.

[32] A. Theodorus, T. K. Prasetyo, R. Hartono, and D. Suhartono, "Short message service (SMS) spam filtering using machine learning in bahasa Indonesia," in *Proc. 3rd East Indonesia Conf. Comput. Inf. Technol. (EIConCIT)*, Indonesia, Apr. 2021, pp. 199–203, doi: 10.1109/EIConCIT50028.2021.9431859.

[33] I. S. Mambina, J. D. Ndibwile, and K. F. Michael, "Classifying Swahili Smishing attacks for mobile money users: A machine-learning approach," *IEEE Access*, vol. 10, pp. 83061–83074, 2022.

[34] T. Sahmoud and D. Mohammad Mikki, "Spam detection using BERT," 2022, *arXiv:2206.02443*.

[35] A. Raza, K. Munir, M. Almutairi, F. Younas, M. M. S. Fareed, and G. Ahmed, "A novel approach to classify telescopic sensors data using bidirectional-gated recurrent neural networks," *Appl. Sci.*, vol. 12, no. 20, p. 10268, Oct. 2022.

[36] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," 2016, *arXiv:1607.01759*.

[37] C. Zhou, C. Sun, Z. Liu, and F. C. M. Lau, "A C-LSTM neural network for text classification," 2015, *arXiv:1511.08630*.

[38] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[39] B. Masua and N. Masasi, "Enhancing text pre-processing for swahili language: Datasets for common swahili stop-words, slangs and typos with equivalent proper words," *Data Brief*, vol. 33, Dec. 2020, Art. no. 106517, doi: 10.1016/j.dib.2020.106517.

[40] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 26, 2013, pp. 1–9.

[41] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global vectors for word representation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1532–1543.

[42] M. S. Jacob and P. Selvi Rajendran, "Fuzzy artificial bee colony-based CNN-LSTM and semantic feature for fake product review classification," *Concurrency Comput., Pract. Exper.*, vol. 34, no. 1, 2022, Art. no. e06539.

[43] S. M. Gowri, G. Sharang Ramana, M. Sree Ranjani, and T. Tharani, "Detection of telephony spam and scams using recurrent neural network (RNN) algorithm," in *Proc. 7th Int. Conf. Adv. Comput. Commun. Syst. (ICACCS)*, vol. 1, Mar. 2021, pp. 1284–1288.

[44] S. Saumya and J. P. Singh, "Spam review detection using LSTM autoencoder: An unsupervised approach," *Electron. Commerce Res.*, vol. 22, no. 1, pp. 113–133, Mar. 2022.

[45] T. C. Alberto, J. V. Lochter, and T. A. Almeida, "TubeSpam: Comment spam filtering on YouTube," in *Proc. IEEE 14th Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2015, pp. 138–143.

IDDI S. MAMBINA received the master's degree in information technology from Punjab Technical University, India, in 2013. He is currently pursuing the Ph.D. degree with The Nelson Mandela-African Institution of Science and Technology.

He joined the University of Dodoma as an Assistant Lecturer. At the university, his core activity is to conduct research, consultancy, and teaching. He assists the senior lecturers in conducting lectures. He also helps students with tutorial sessions with the Department of Information Systems and Technology. His research interests include the spectrum of natural language processing, social engineering, cyber security, the psychology of social engineering, data science, information technology for development, and artificial and human intelligence. His current research interest includes applying natural language techniques to social engineering attacks.

JEMA D. NDIBWILE received the Engineering Ph.D. degree in information security from the Nara Institute of Science and Technology, Japan, in 2019.

He is currently an Assistant Professor in cybersecurity with Carnegie Mellon University Africa. Assisting in addressing complex cyber security challenges, his specialization includes cybersecurity, military intelligence, applied cryptography, ethical hacking, the psychology of cybersecurity, digital forensics, and cyber defenses. His current research interests include usable privacy and security, hacking countermeasures, the impact of artificial and human intelligence on cybersecurity, and social engineering approaches. He has extensive experience in ethical hacking/penetration testing, digital forensics, and project management leveraging tools, such as Kali Linux, Parrot OS, and Cellebrite.

DEO UWIMPUHWE received the bachelor's degree in electrical power engineering, in 2016. He is currently pursuing the Master of Science degree in electrical and computer engineering (MSECE) and majoring in applied machine learning with Carnegie Mellon University (CMU).

He has a strong background in electrical engineering. With five years of professional experience in the energy sector, he has noticeable project management skills and a substantial understanding of power system necessities, especially power substations. His extensive research interests include the application of deep learning in remote sensing technology and other computer vision and natural language processing-based applications, energy data collection and analysis, and the impact of mobile big data in smart city development. His key experience, particularly, is in project management and the quality assurance of electrical power system projects; he is also familiar with using Python in different applications of machine learning.

KISANGIRI F. MICHAEL received the Ph.D. degree in the field of telecommunications from the Wroclaw University of Technology, Poland.

He has been with the School of Computation and Communication Science and Engineering, The Nelson Mandela-African Institution of Science and Technology (NM-AIST), as a Lecturer and then a Senior Lecturer, since December 2011. Before joining NM-AIST, he was with the Dar es Salaam Institute of Technology as a Lecturer for three years. Currently, he is an Academician and has supervised dozens of M.Sc. and several Ph.D. research. He possesses the good knowledge of artificial intelligence, antenna design, and wireless communication systems. He is a fluent speaker of three languages, such as Swahili, English, and Polish.