IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 28 July 2023, accepted 20 August 2023, date of publication 24 August 2023, date of current version 7 September 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3308515

RESEARCH ARTICLE

Deepfake Detection on Social Media: Leveraging Deep Learning and FastText Embeddings for Identifying Machine-Generated Tweets

SAIMA SADIQ[1], TURKI ALJREES[2], AND SALEEM ULLAH[1]
[1]Department of Computer Science, Khwaja Fareed University of Engineering and Information Technology, Rahim Yar Khan 64200, Pakistan
[2]Department of Computer Science and Engineering, University of Hafr Al Batin, Hafar Al Batin 39524, Saudi Arabia

Corresponding author: Saleem Ullah (Saleem.ullah@kfueit.edu.pk)

This work was supported by the University of Hafr Al Batin and the Department of Computer Science, Khwaja Fareed University of Engineering and Information Technology, Rahim Yar Khan, Pakistan.

ABSTRACT Recent advancements in natural language production provide an additional tool to manipulate public opinion on social media. Furthermore, advancements in language modelling have significantly strengthened the generative capabilities of deep neural models, empowering them with enhanced skills for content generation. Consequently, text-generative models have become increasingly powerful allowing the adversaries to use these remarkable abilities to boost social bots, allowing them to generate realistic deepfake posts and influence the discourse among the general public. To address this problem, the development of reliable and accurate deepfake social media message-detecting methods is important. Under this consideration, current research addresses the identification of machine-generated text on social networks like Twitter. In this study, a simple deep learning model in combination with word embeddings is employed for the classification of tweets as human-generated or bot-generated using a publicly available Tweepfake dataset. A conventional Convolutional Neural Network (CNN) architecture is devised, leveraging FastText word embeddings, to undertake the task of identifying deepfake tweets. To showcase the superior performance of the proposed method, this study employed several machine learning models as baseline methods for comparison. These baseline methods utilized various features, including Term Frequency, Term Frequency-Inverse Document Frequency, FastText, and FastText subword embeddings. Moreover, the performance of the proposed method is also compared against other deep learning models such as Long short-term memory (LSTM) and CNN-LSTM displaying the effectiveness and highlighting its advantages in accurately addressing the task at hand. Experimental results indicate that the design of the CNN architecture coupled with the utilization of FastText embeddings is suitable for efficient and effective classification of the tweet data with a superior 93% accuracy.

INDEX TERMS Text classification, machine learning, deep learning, deepfake, machine generated text.

I. INTRODUCTION

Social media platforms were created for people to connect and share their opinions and ideas through texts, images, audio, and videos [1]. A bot is computer software that manages a fake account on social media by liking, sharing, and uploading posts that may be real or forged using techniques like gap-filling text, search-and- replace, and video editing or deepfake [2]. Deep learning is a part of machine learning that learns feature representation from input data. Deepfake is a combination of ''deep learning'' and ''fake'' and refers to artificial intelligence-generated multimedia (text, image, audio and video) that may be misleading [3]. Deepfake multimedia's creation and sharing on social media have already

The associate editor coordinating the review of this manuscript and approving it for publication was Jon Atli Benediktsson.

created problems in a number of fields such as politics [4] by deceiving viewers into thinking that they were created by humans.

Using social media, it is easier and faster to propagate false information with the aim of manipulating people's perceptions and opinions especially to build mistrust in a democratic country [5]. Accounts with varying degrees of humanness like cyborg accounts to sockpuppets are used to achieve this goal [6]. On the other hand, fully automated social media accounts also known as social bots mimic human behaviour [7]. Particularly, the widespread use of bots and recent developments in natural language-based generative models, such as the GPT [8] and Grover [9], give the adversary a means to propagate false information more convincingly. The Net Neutrality case in 2017 serves as an illustrative example: millions of duplicated comments played a significant role in the Commission's decision to repeal [10]. The issue needs to be addressed that simple text manipulation techniques may build false beliefs and what could be the impact of more powerful transformer-based models. Recently, there have been instances of the use of GPT-2 [11] and GPT-3 [12]: to generate tweets to test the generating skills and automatically make blog articles. A bot based on GPT-3 interacted with people on Reddit using the account ''/u/thegentlemetre'' to post comments to inquiries on /r/AskReddit [13]. Though most of the remarks made by the bot were harmless. Despite the fact that no harm has been done thus far, OpenAI should be concerned about the misuse of GPT-3 due to this occurrence. However, in order to protect genuine information and democracy on social media, it is important to create a sovereign detection system for machine-generated texts, also known as deepfake text.

In 2019, a generative model namely GPT-2 displayed enhanced text-generating capabilities [12] which remained unrecognizable by the humans [14], [15]. Deepfake text on social media is mainly written by the GPT model; this may be due to the fact that the GPT model is better than Grover [16] and CTRL [17] at writing short text [18]. Consequently, it is highly challenging to detect machine-generated text produced by GPT-2 than by RNN or other previously generated techniques [19]. To address this significant challenge, the present study endeavours to examine deepfakes generated by RNN, as well as GPT-2 and various other bots. Specifically, the study focuses on employing cutting-edge deepfake text detection techniques tailored to the dynamic social media environment. State-of-the-art research works regarding deepfake text detection include [15], [19], [20]. Authors in [21] improved the detection of deepfake text generated by GPT 2.

Deepfake detecting techniques are constantly being improved, including deepfake audio identification techniques [22], [23], deepfake video screening methods [24], and deepfake text detection techniques. Neural network models tend to learn characteristics of machine-generated text instead of discriminating human-written text from machine text [25]. Some techniques like replacing letters with homoglyphs and adding commonly misspelled words have made the machine-generated text detection task more challenging [25]. In addition, previous studies mostly performed deepfake text detection in long text-like stories and news articles. The research claimed that it is easier to identify deepfakes in longer text [26]. The use of cutting-edge detection methods on machine-generated text posted on social media is a less explored research area [26]. Text posted on social media is often short, especially on Twitter [27]. There is also a lack of properly labelled datasets containing human and machine-generated short text in the research community [19]. Researchers in [28] and [29] used a tweet dataset containing tweets generated by a wide range of bots like cyborg, social bot, spam bot, and sockpuppet [30]. However, their dataset was human labelled and research claimed that humans are unable to identify machine-generated text. The authors in [19] provided a labelled dataset namely Tweepfake containing human text and machine-generated text on Twitter using techniques such as RNN, LSTM, Markov and GPT-2. With the aim of investigating challenges faced in the detection of deepfake text, this study makes use of the same dataset.

The dataset containing both bot-generated and human-written tweets is used to evaluate the performance of the proposed method. This study employs various machine learning and deep learning models, including Decision Tree (DT), Logistic Regression (LR), AdaBoost Classifier (AC), Stochastic Gradient Descent Classifier (SGC), Random Forest (RF), Gradient Boosting Machine (GBM), Extra tree Classifier (ETC), Naive Bayes (NB), Convolutional Neural Network (CNN), Long Short Term Memory (LSTM), and CNN-LSTM, for tweet classification. Different feature extraction techniques, such as Term Frequency (TF), Term frequency-inverse document frequency (TF-IDF), FastText, and FastText subwords are also explored to compare their effectiveness in identifying machine-generated text. This research provides the following contributions:

- Presenting a deep learning framework combined with word embeddings that effectively identifies machine-generated text on social media platforms.
- Comprehensive evaluation of various machine learning and deep learning models for tweet classification.
- Investigation of different feature extraction techniques for detecting deepfake text, with a focus on short text prevalent on social media.
- Demonstrating the superiority of our proposed method, incorporating CNN with FastText embeddings, over alternative models in accurately distinguishing machine-generated text in the dynamic social media environment.

The rest of the article is structured as follows: Section II discusses past work on deepfake text identification, and Section III presents deepfake generation methods. Section IV outlines the material and methods used in experiments to enhance deepfake tweet detection, and Section V describes the result and discussion section. Section VI presents and discusses the findings.

II. RELATED WORK

Deepfake technologies initially emerged in the realm of computer vision [31], [32], [33], advancing towards effective attempts at audio manipulation [34], [35] and text synthesis [36]. In computer vision, deepfakes often involve face manipulation, including whole-facial synthesis, identity swapping, attribute manipulation, and emotion switching [22]-as well as body reenactment [37]. Audio deepfakes, which have recently been used, generate spoken audio from a text corpus using the voices of several speakers after five seconds of listening [34]. The upgrading of the language models was made possible in 2017 because of the development of the self-attention mechanism and the transformer. Language modelling estimates the likelihood that a given sequence of words will appear in a sentence using various statistical and probabilistic methodologies. The succeeding transformer-based language models (GPT [38], BERT [39], GPT-2 [36], etc.) improved not only language-generating tasks but also natural language interpretation tasks. In 2019, Radord et al. [36] created GPT-2, a pre-trained language model that can create paragraphs of text that are coherent and human-like on their own with just one short sentence as input. The same year, authors [9] developed GROVER, a novel method for quickly and effectively learning and creating multi-field documents like journal articles. The conditional language model CTRL, which employs control codes to produce text with a particular style, content, and task-specific behaviour, was published shortly after [17]. Researchers [40] introduced OPTIMUS which included a variational autoencoder in the text production process.

The GPT-2 research team conducted an internal detection study [41] using text samples generated by the GPT-2. First, they assessed a conventional machine-learning method that trains a logistic regression discriminator on TF-IDF unigram and bigram characteristics. Following that, they tested a simple zero-shot baseline using an overall probability threshold: a text excerpt is classified as machine-generated if, according to GPT-2, its likelihood is closer to the mean likelihood over all machine-generated texts than to the mean of human-written texts.

Giant Language Model Test Room (GLTR), is a visual tool that aids people in spotting deepfake texts [42]. The generated text is sampled word per word from a next token distribution; this distribution typically differs from the one that people unconsciously use when they write or speak (many sampling approaches may be employed, but the simplest option is to take the most probable token). In order to help individuals distinguish between human-written text samples and machine-generated ones, GLTR aims to display these statistical linguistic distinctions.

By employing BERT, GPT2, and GROVER as the pre-trained language model, the developers of GROVER [9] adhered to the fine-tuning-based detection methodology. GROVER was the best, with the argument that a detector with a similar architectural design may be the greatest line of defence against text generators that use transformers. On GPT-2 generated texts, however, OpenAI [41] disproved it by demonstrating that fine-tuning a RoBERTa-based detector consistently produced greater accuracy than fine-tuning a GPT-2-based detector with comparable capacity. Unlike auto-regressive language models (such as GPT-2 and XLNET [43]), which are defined in terms of a series of conditional distributions, authors [44] created an energy-based deepfake text detector. An energy-based model is defined in terms of a single scalar energy function, which represents the joint compatibility between all input variables. The deepfake discriminator is an energy function that evaluates the joint compatibility of a series of input tokens given some context (such as a text sample, a few keywords, a collection of phrases, or a title) and a set of network parameters. The experimental context, where the text corpora and generator designs alter between training and testing, was also attempted to generalise by these authors.

Authors in [15] carried out the sole study on identifying deep-fake social media messages on GPT-2-based Amazon evaluations. The Grover-based detector, GLTR, the RoBERTa-based detector from OpenAI, and a straightforward ensemble that combined these detectors using logistic regression at the score level were among the human-machine discriminators that were assessed. The aforementioned deepfake text detection techniques have two drawbacks: aside from the study [15], they focused on creating news stories, which are lengthier than social media communications. Additionally, only one known adversarial generating model is often used to produce deepfake text samples (GPT-2 or GROVER). We are not sure about the number and type of generative architectures employed in a real-world scenario.

Existing research in deepfake text detection includes methods like graph-based approach [45], feature-based approach [46], and deep learning models like BiLSTM [47] and RoBERTa [19]. In a survey [48], the researchers offered a more profound insight into the creation and identification of deepfakes, the prevailing patterns and progressions in this field, and the limitations of existing defence mechanisms. These studies have focused on creating and detecting news stories, which are typically longer than social media communications. This raises concerns about the generalizability of such methods to the specific challenges posed by short text on social media. Some studies [47], [49] used the PAN dataset which focuses on determining profiles of fake accounts. Others [46], [50], [51] used the Cresci dataset, which used profile features like tweet content, activity patterns, and network characteristics to find bot accounts. To aid the research community in identifying shorter deepfake texts created by different generating approaches, our Tweepfake dataset offers a collection of tweets generated by several generative models.

TABLE 1. Comparative analysis of the existing approaches.

Ref.	Year	Methods	Dataset	Findings
[50]	2018	LSTM	Cresci and collaborators dataset	They employed tweet level bot detection using textual features and metadata. Their proposed architecture achieved 96% AUC score in bot detection.
[52]	2019	Deep Forest Algorithm	Dataset collected from the Twitter time-line(API)	Authors applied SMOTE in combination with proposed model and outperformed machine learning models.
[53]	2019	Fine-tuned BERT	Create their dataset using GPT-2 model	Authors investigated machine-generated text using discrminators on balanced dataset.
[49]	2020	BERT-based	English tweets from the PAN competition dataset	Authors designed a bot detection model and their proposed model achieved 83.36% weighted F1-score.
[54]	2020	Bot-AHGCN	CTU-13 dataset and their own collected botnet dataset	A multi-attributed heterogeneous graph convolution approach has been applied for bot
[55]	2020	Gaussian kernel density peak clustering algorithm (GKDPCA)	dataset consisting 1971 normal human accounts and 462 social bot accounts	Oversampling techniques are applied to improve the classification.
[19]	2021	RoBERTa based detector	TweepFake dataset	Authors discriminated human written text from bot text and presented deepfake tweet dataset and also applied 13 models for deepfake detection.
[47]	2022	BiLSTM	A dataset consisting 30000 tweets from PAN-20	Authors investigate role of social bots in spreading fake news
[51]	2022	Google Bert	Cresci 2017 data set	Classified dataset into bot or human written with 94% accuracy.
[56]	2022	GANBOT framework	Twitter social bot	Their proposed model outperformed other previous contextual LSTM methods for bot detection.
[45]	2022	Graph based approach	9 datasets including TwiBot-22	Authors addressed graph based detection on large scale.
[46]	2022	Feature-based approach	5 Datasets (Cresci2015, Cresci2017, Lee2011, Moghaddam2019, and Varol2017)	Friendship preference features extracted from profiles are employed for Bot detection.
[57]	2023	Logistic Regression, RoBERTa	Human ChatGPT Comparison Corpus (HC3)	Authors performed human analysis, linguistic evaluation and bot-generated text detection and provided deep insight to future research.
[58]	2023	XGBoost	Human-written essays and ChatGPT generated essays	Authors investigated TF-IDF and hand-crafted features to detect ChatGPT generated text.
[59]	2023	Transformer-based ML Model (DistilBERT)	ChatGPT query Dataset, ChatGPT rephrase Dataset	Authors explored that identifying ChatGPT text of rephrasing is more challenging. They also provide a deep insight into the writing style of ChatGPT.
[60]	2023	Fine-tune-based detection model (RoBERTa)	Human-written abstracts and AI-generated abstracts	Authors investigated gap between machine-generated text and human-written text.

III. DEEPFAKE TEXT GENERATION METHODS

There are several ways to create deepfake text. Here is a brief explanation of some common generative techniques used to create computer-generated text.

Markov Chains is a stochastic model that depicts a succession of states by transitioning from one state to another with a probability that solely depends on the current state. State tokens are used in the text creation, and the next token/state is chosen at random from a list of tokens after the current one. The frequency with which a token follows the current token, t, determines the likelihood that token t will be picked.

The RNN computes the multinomial distribution from which the next token will be picked, and with the aid of its loop structure, saves the knowledge about the previously encountered tokens in its accumulated memory. The chosen token is returned as input so that the RNN can generate the next one.

As a sampling strategy, the RNN+Markov method may use the Markov Chain's next token selection. In practice, the next token is drawn at random from the RNN-generated multinomial distribution, with the tokens with the greatest probability value being the most likely to be picked. However, no references were discovered to support our RNN+Markov mechanism theory.

LSTM creates text in the same way as RNN does. However, because of its more sophisticated structure, it is wiser than the latter: it can learn to selectively maintain track of just the necessary information of previously viewed text while simultaneously reducing the vanishing gradient problem that concerns RNNs. The memory of an LSTM is "longer" than that of an RNN.

GPT-2 is a generative pre-trained transformer language model based on the Attention mechanism: by using Attention, a language model pre-trained on millions of sentences/texts learns how each token/word connects to every other in every conceivable situation. This is the method for creating more cohesive and non-trivial paragraphs of text. As a language model, GPT-2's text generation processes are the same as RNN and LSTM: production of a multinomial distribution at each step, followed by selection of the next token from it using a specific sampling strategy.

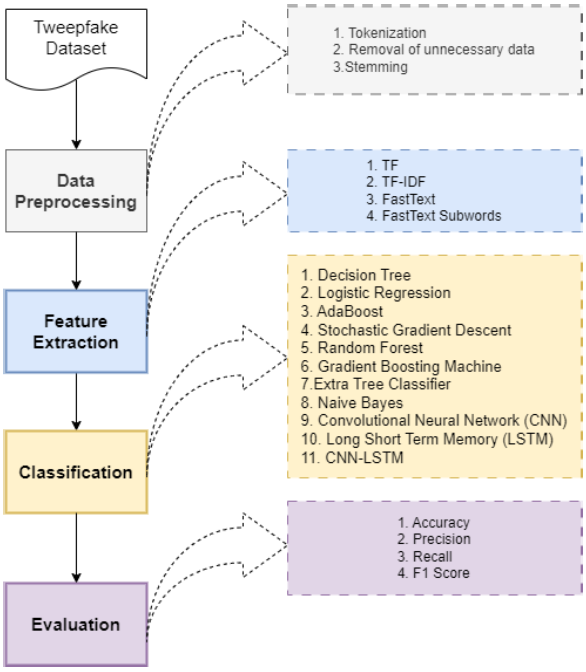

FIGURE 1. Architecture of methodologies adopted for deepfake tweet classification.

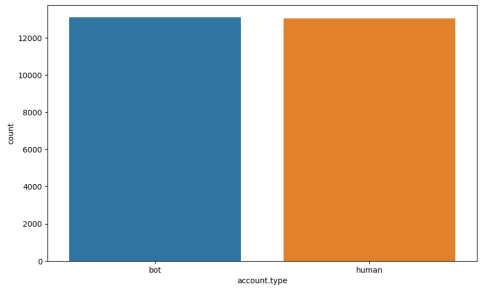

FIGURE 2. Countplot showing account-type data distribution.

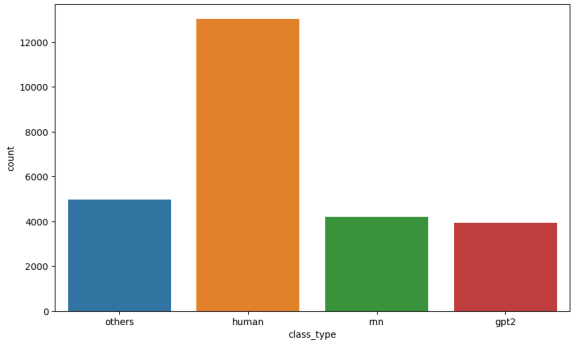

FIGURE 3. Countplot showing class-wise data distribution.

The third-generation Generative Pre-trained Transformer (GPT-3) is a neural network machine learning model that can produce any type of text. It was trained using internet data. It was created by OpenAI and uses a tiny quantity of input text to produce vast quantities of accurate and complex machine-generated text.

IV. MATERIAL AND METHODS

This section discusses the dataset, feature engineering techniques, machine learning models and deep learning models used in the experiments. The experimental strategy is presented in Figure 1.

A. DATASET

This study utilizes TweepFake [19] dataset containing 25572 tweets in total. The dataset comprises tweets from 17 human accounts and 23 bot accounts. Each count of human and bot is properly labelled. The latter identifies the text creation method that was used, which might be human (17 accounts, 12786 tweets), GPT-2 (11 accounts, 3861 tweets), RNN (7 accounts, 4181 tweets), or Others (5 accounts, 4876 tweets). Figure 2 presents the Count-plot showing account-type data distribution and Figure 3 shows the Count-plot showing class-wise data distribution.

1) DATA PRE-PROCESSING

Datasets include useless data in unstructured or semi-structured form. Such unnecessary data lengthens the model's training time and may degrade its performance. Pre-processing is critical for increasing the efficiency of ML models and conserving computing resources. Preparing the text improves the model's ability to anticipate outcomes accurately. Pre-processing includes the following steps: tokenization, case conversion, stopword removal, and removal of numbers.

Due to the case sensitivity of machine learning models, the model will treat the occurrence of the terms "MACHINE" and "machine" as two different words. As a result, the dataset must first be converted into lowercase as part of the preprocessing.

The second preparation phase is removing hashtags and usernames from tweets. The data is cleaned off of punctuation such as %##&().,'" These punctuations have a direct impact on performance since they make it harder for algorithms to tell these symbols apart from textual terms.

The next step is to do stopword removal. Stopwords make sentences easier for people to read, but classification algorithms cannot understand them.

B. FEATURE EXTRACTION

To train the machine learning models, feature extraction techniques are necessary. The models are trained in this study using the following feature extraction methods.

1) TF

The term frequency (TF) measures the frequency with which a term (word or token) appears in a text or corpus of documents. It is a simple yet basic idea in both machine

learning and natural language processing. Different text-based machine learning algorithms frequently employ TF as a feature to assess the significance or relevance of a phrase inside a document. It is determined by counting how many times a phrase appears in a document and then normalising that figure according to the total number of terms in the text. The idea behind TF is that a term's likelihood to be relevant or suggestive of the content of a document increases with the frequency with which it appears in the text.

2) TF-IDF

Another frequently used method for feature extraction from unprocessed text data is TF-IDF. The majority of its applications are in text categorization and information retrieval [61]. In contrast to TF's basic term count, TF-IDF additionally gives each word a weight based on how important it is. Inverse document frequency and word frequency are used in this process [62].

$$W_{i,j} = [1 + log(tf_{i,j})] \times [log(\frac{N}{df_i})] \quad (1)$$

where N refers to the total number of documents, $TF_{i,j}$ stands for term frequency inside a document, and$D_{f,t}$ refers to the document carrying the text/word t.

C. WORD EMBEDDING TECHNIQUES

1) FastText

FastText is a free and open-source toolkit developed by Facebook AI Research (FAIR) for learning word embeddings and classifications. It has 600 billion worth of word vectors and 2 million popular crawl words with 300−dimensions. Along with single words, it employs hand-crafted n-grams as features. Because of its simple architecture, text categorization is conducted extremely effectively and efficiently [63]. This approach enables the development of unsupervised or supervised learning algorithms for producing word vectors. It can train enormous datasets in minutes and is useful for detecting semantic similarities and text categorization.

Various text categorization problems have made use of various word embedding approaches. Pre-trained word embeddings anticipate the context of the words in an unsupervised way. Words that are close together are seen as having a similar context. FastText embedding is a good option for representing vectors since it employs morphological cues to identify challenging words. Its generalizability is enhanced by this capability. FastText word embedding uses n-grams to create vectors, which aids in handling words that are unknown.

2) FastText SUBWORD

FastText Subword has 2 million word vectors that were learned using Common Crawl's subword information (600B tokens). By breaking down each word into its component words, subword embedding gives us more information [64]. FastText can create word representations that are capable of encapsulating the meaning of individual morphemes or smaller parts of words by taking into account subword information. This is especially helpful for languages with complex morphology, where words might take on several forms depending on the present or past tense, plural forms, gender, etc. The way FastText handles subwords is as follows:

- FastText divides words into character n-grams and then creates a vocabulary from the input corpus. It takes into account both the whole word and its component parts.
- The sum of a word's character n-gram embeddings serves as its representation. The word embeddings and the character n-gram embeddings are both learned during the training procedure.
- FastText takes into account subword units, hence it can create representations for Out-of-vocabulary words by merging the character n-gram embeddings of those words. FastText can now offer meaningful representations for words that were not present during training.
- Text categorization tasks may be performed with FastText. To anticipate the labels of incoming text, it trains a classifier on top of word representations.

D. DEEP LEARNING MODELS

This section presents the deep learning models employed in the experiments including CNN, LSTM and CNN-LSTM. The layered architecture of the deep learning models is presented in Table 4.

1) CNN

A CNN is a deep neural network whose pooling and convolution layers both learn sophisticated features. The majority of the time, CNN is employed for jobs involving picture classification and segmentation. The layered CNN model is more reliable since it has undergone end-to-end training. Since this is a feed-forward network model, features are mapped by using filters on the output of the layers. The CNN model also includes layers that are fully connected, drop out, and pooling layers. The fully connected layers get input from the output of the preceding levels to determine the outcome. Pooling layers-which may be either maximum or average-play a part in feature selection. In the eq 2, the ReLU function is used as an activation function.

$$y = max(0, i) \quad (2)$$

where y represents the activation's output and i represents its input. Convolutional layers use weight to extract high-level features for training. Cross entropy is a loss function that is calculated according to the formula shown in equation 8.

$$crossEntropy = -(i\ log(p) + (1 - i)\ log(1 - p)) \quad (3)$$

where p is the expected probability and i displays class labels. The Sigmoid error function is used to predict CNN output

TABLE 2. Machine learning models.

Reference	Model	Description
[65]	RF	The RF classifier is a decision tree-based system that employs numerous weak learners to make very accurate predictions. RF uses a technique known as 'bootstrap bagging' to train multiple decision trees using various bootstrap samples. A bootstrap sample is generated by randomly subsampling the training dataset, which has the same size as the test dataset. RF, like other ensemble classifiers, makes predictions using decision trees. Choosing the root node at each level of decision tree construction might be difficult.
[66]	DT	DT is a well-known machine learning technique with several applications in regression and classification issues. At each level of the DT, picking the root node, also known as 'attribute selection,' is a vital element of the process. The 'Gini index' and 'information gain' are both well-known attribute selection approaches. The Gini index is often used to determine the extent of impurity in a dataset.
[67]	LR	Because of its use of the logistic equation (also known as the sigmoid function), LR is a popular approach for dealing with binary classification issues. This function converts every given numerical value into a number between 0 and 1 using an S-shaped curve, which is why LR is so popular.
[68]	SGC	It is a popular optimization algorithm that updates the model parameters by minimizing the loss function using small batches of training data. This algorithm works in an iterative manner for computational efficacy. Consequently, the frequent updation of parameters leads to faster convergence hence making this algorithm the best fit for larger datasets.
[69]	AC	It combines multiple weak learners to develop a strong classifier. It is an iterative algorithm that assigns higher weights to incorrectly classified samples in each iteration, enabling the subsequent weak learners to focus on challenging examples. The final classification is made by aggregating the predictions from all weak learners. AC is well-known for its capability to deal with complex patterns in the dataset and enhance the classification accuracy with each iteration.
[70]	GBM	GBM is an effective ensemble learning technique that merges numerous weak prediction models, such as decision trees, to construct a robust classifier. It sequentially trains the models, with each subsequent model learning from the errors made by its predecessors. The GBM algorithm enhances a loss function by iteratively incorporating models that minimize the loss gradient. This iterative approach results in the development of a powerful model capable of accurately predicting the target variable.
[71]	NB	It is a widely used probabilistic machine learning algorithm that is based on Bayes' theorem. NB assumes that features are independent of each other given the class label, which simplifies the calculation process. By calculating the probabilities of each class for a given input, it chooses the class with the highest probability as the predicted class. Naive Bayes is known for its simplicity, fast training and prediction speed, and its effectiveness in handling high-dimensional data.
[72]	ETC	ETC is a well-known ensemble learning approach for making accurate predictions by combining many decision trees. In contrast to RF, ETC chooses a subset of the best characteristics at random for each split in the decision tree. This approach produces de-correlated trees, which are less sensitive to individual attributes and more resilient to noise. ETC selects the appropriate feature to partition the data based on the Gini index. It also assesses the significance of traits according to their Gini score.

TABLE 3. Hyperparamter tuning values of machine learning models.

Model	Hyperparameters
DT	Trees=200, max_depth=30, random state=52
LR	penalty='l2', solver='lbfgs'
AC	n_estimators=300, max_depth=300, lr=0.2
SGC	alpha=1.0, binarize=0.0
RF	Trees=200, max_depth=30, random state=52
GBM	Trees=200, max_depth=30, random state=52, lr=0.1
ETC	Trees=200, max_depth=30, random state=52
NB	alpha=1.0, binarize=0.0

since it is an enhanced form of the backpropagation model. Two target classes are output by the CNN model.

2) LSTM

LSTM is one of the state-of-the-art deep learning methods that are frequently employed to address text classification issues. Three gates make up an LSTM: the input gate (ik), the output gate (ok), and the forget gate (fk). When data passes through these gates, the gates retain critical information while forgetting irrelevant information based on dropout value. An important piece of data is stored in the memory block Ck. There are several LSTM variations; the one utilised in this study is provided in eq. 9, 10 and 11.

$$i_k = \sigma(W_i s_k + V_i h_{k-1} + b_i) \tag{4}$$
$$f_k = \sigma(W_f s_k + V_f h_{k-1} + b_f) \tag{5}$$
$$o_k = \sigma(W_o s_k + V_o h_{k-1} + b_o) \tag{6}$$
$$c_k = tanh(W_c x_k + V_c h_{k-1} + b_c) \tag{7}$$

The corresponding weights for the matrix members are represented by W and V. While sk displays the input of a certain time and b displays the bias, h displays the hidden state up to $k1$ time steps. Memory block cell c is refreshed every $k1$ time step. Every neuron in the dense layer is coupled to every other neuron in the output layer of the LSTM.

3) CNN-LSTM

Compared to individual models, ensemble models frequently exhibit superior performance. The advantages of combining CNN's powerful automated feature extraction with LSTM's capacity to capture long-term temporal dependencies have led to the widespread adoption of the ensemble of CNN and LSTM. Providing correct feature representations enables the

LSTM layers to learn temporal connections. CNN-LSTM is an efficient solution for dealing with time series and classification issues [73].

TABLE 4. Layered structure of deep learning models.

CNN	LSTM	CNN-LSTM
Conv (7 × 7, @64)	LSTM (100 neurons)	Conv (7 × 7, @64)
Max pooling (2 × 2)	Dropout (0.5)	Max pooling (4 × 4)
Conv (7 × 7, @64)	Dense (32 neurons)	LSTM (100 neurons)
GlobalMax pooling (2 × 2)	Softmax (2)	Dense (32 neurons)
Dropout (0.5)		Softmax (2)
Dense (32 neurons)		
Softmax (2)		

E. PROPOSED METHODOLOGY

This section presents the proposed methodology adopted for tweet classification. The architecture of the proposed framework is presented in Figure 4. Deep learning models like CNN can automatically learn significant features from text input. They are capable of capturing hierarchical patterns, local relationships, and long-term connections, allowing the model to extract usable representations from the incoming text. By stacking multiple layers of CNN, dependencies of text can be captured. This work introduces a simple deep learning-based CNN model for tweet classification.

In the proposed framework, a labelled dataset is collected from a public repository. The collected dataset contains tweets from human and bot accounts. In order to simplify the text and enhance its quality, a series of preprocessing steps are employed to clean the tweets. The dataset is divided into 80:20 ratios for training and testing. The next step involves transforming the text into vectors using FastText word embedding. Subsequently, these vector representations are fed into the CNN model. The proposed methodology, which leverages FastText word embedding in conjunction with a 3-layered CNN, is employed for the training process. The efficacy of this approach is assessed through the utilization of four evaluation metrics: Accuracy, Precision, Recall, and F1-score.

F. EVALUATION METRICS

The evaluation of models involves assessing their performance using several widely utilized measures to classify deep fake text. These measures include accuracy, recall, precision, and F1 score. Accuracy assesses a model's overall accuracy by measuring the ratio of properly categorized examples to the total number of occurrences. It provides a general overview of the model's performance.

$$Accuracy = \frac{\text{Number of correctly classified predictions}}{\text{Total predictions}} \tag{8}$$

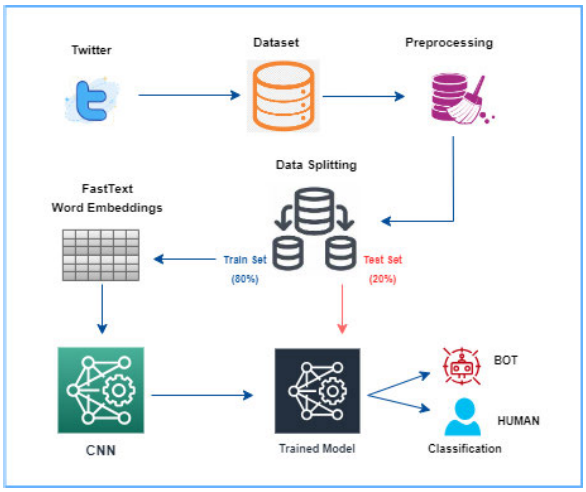

FIGURE 4. Architecture of proposed framework for deepfake tweet classification.

while in the case of binary classification, accuracy is measured as:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{9}$$

whereas TP is true positive, FP is false positive, TN is true negative, and FN is false negative and can be defined as.

TP: TP represents the positive predictions of a correctly predicted class.

FP: FP represents the negative predictions of an incorrectly predicted class.

TN: TN represents the negative predictions of a correctly predicted class.

FN: FN represents the positive predictions of an incorrectly predicted class

Precision represents the model's accuracy in categorizing positive cases as the ratio of successfully categorised positive instances to total positive instances anticipated. The model's ability to avoid false positives is highlighted.

$$Precision = \frac{TP}{TP + FP} \tag{10}$$

Recall, also known as sensitivity or true positive rate, assesses the model's ability to properly identify positive events out of all real positive instances. It is especially essential in situations when the identification of positive cases is critical, such as illness diagnosis.

$$Recall = \frac{TP}{TP + FN} \tag{11}$$

The F1 score gives a fair assessment of a model's performance by taking accuracy and recall into consideration. It is scored on a scale of 0 to 1, with a higher score signifying better performance. A flawless F1 score of 1 indicates that the model achieved perfect accuracy as well as recall. To summarise, the F1 score combines accuracy and recall into a

single statistic that provides an overall evaluation of a model's performance in classification tasks.

$$F1score = 2 \times \frac{precision \times recall}{precision + recall} \quad (12)$$

V. RESULTS AND DISCUSSION

This section describes the experiment that was carried out for this study and discusses the outcomes. In this study, machine learning and deep learning models are employed to detect deepfake tweets. To validate the proposed approach, this study employs eight machine learning models: DT, LR, AC, SGC, RF, GBM, ETC and NB. The description of these models is presented in Table 2. These models are employed using the hyperparameters that are best suited to the dataset. Value ranges are fine-tuned to provide the greatest performance when selecting the optimum hyperparameters. Table 3 shows the hyperparameter settings and tuning range.

A. EXPERIMENTAL SETUP

The results of several classifiers for identifying deepfake text are presented in this section. The models were developed using Python 3.8 and a Jupyter Notebook, and the testing was carried out on a device running Windows 10 and a 7th-generation Core i7 CPU. Accuracy, precision, recall, and F1 score were used to evaluate how well the learning models performed. Table 5 provides detailed information on the hardware and software requirements used in the experiment.

TABLE 5. Experimental setup for the proposed system.

Element	Details
Language	Python 3.8
OS	64-bit window 10
RAM	8GB
GPU	Nvidia, 1060, 8GB
CPU	Core i7, 7th Gen with 2.8 GHz processor

B. PERFORMANCE OF MACHINE LEARNING MODELS

This section details the experiments conducted in this research and a discussion of obtained results. The experimental results utilizing various feature engineering strategies are examined for deepfake text detection. The frequency-based techniques (TF and TF-IDF) and Word Embedding techniques (FastText and FastTest Subword) are used to compare the output of various supervised machine learning models. Machine learning models including DT, LR, AC, SGC, RF, GB, ETC and NB are compared in terms of accuracy, precision, recall and F1 score. The effectiveness of every model varies depending on the feature extraction method.

1) COMPARISON OF CLASSIFIERS USING TF

The performance comparison of machine learning models is presented in Table 6. Results show that ETC using TF has demonstrated superior performance with a 0.83 value of accuracy, precision, recall and F1 Score. Similarly, RF and NB have attained 0.83 values of accuracy and recall in classifying deepfake text. DT, LR, AC, SGC and GBM have shown similar results with 0.75 accuracy, 0.80 precision, 0.75 recall and 0.77 F1 score.

TABLE 6. Classification result using TF.

Model	Accuracy	Precision	Recall	F1-Score
DT	0.75	0.80	0.75	0.77
LR	0.75	0.80	0.75	0.77
AC	0.75	0.80	0.75	0.77
SGC	0.75	0.80	0.75	0.77
RF	0.83	0.69	0.83	0.76
GBM	0.75	0.80	0.75	0.77
ETC	0.83	0.83	0.83	0.83
NB	0.83	0.69	0.83	0.76

2) COMPARISON OF CLASSIFIERS USING TF-IDF

Table 7 presents the result of machine learning models using TF-IDF for classifying deepfake text. It can be observed that the use of TF-IDf has significantly improved the results of RF and ETC. Both models achieved 0.91 scores in accuracy, precision, recall and F1 score which is the highest score among classifiers using TF-IDF. DT, AC, SGC, GBM and NB did not show any improvement as compared to the results obtained with TF.

TABLE 7. Classification result using TF-IDF.

Model	Accuracy	Precision	Recall	F1-Score
DT	0.75	0.80	0.75	0.77
LR	0.83	0.69	0.83	0.76
AC	0.75	0.80	0.75	0.77
SGC	0.75	0.80	0.75	0.77
RF	0.91	0.91	0.91	0.91
GBM	0.75	0.80	0.75	0.77
ETC	0.91	0.91	0.91	0.91
NB	0.83	0.69	0.83	0.76

3) COMPARISON OF CLASSIFIERS USING FastText

Using FastText embedding for deepfake text, the efficiency of supervised machine learning models is also compared. FastText embedding has shown to be an excellent text classification tool. The experimental results in Table 8 reveal that when applied with FastText, the supervised machine learning models do not produce robust results. ETC and RF yield the greatest result with an accuracy of 0.90 which is lower than the 0.91 obtained by utilizing TF-IDF features. These results highlight that the FastText embedding technique does not enhance the effectiveness of any classifier, as evidenced by the experimental outcomes.

4) COMPARISON OF CLASSIFIERS USING FastText SUBWORD

The efficacy of FastText Subword is also investigated in combination with machine learning models or deep fake text detection. While classifying tweets using FastTest Subword, it can be observed that the performance of machine learning

TABLE 8. Classification result using FastText.

Model	Accuracy	Precision	Recall	F1-Score
DT	0.75	0.80	0.75	0.77
LR	0.83	0.69	0.83	0.76
AC	0.75	0.80	0.75	0.77
SGC	0.75	0.80	0.75	0.77
RF	0.90	0.90	0.90	0.90
GBM	0.75	0.80	0.75	0.77
ETC	0.90	0.90	0.90	0.90
NB	0.83	0.69	0.83	0.76

models has degraded significantly as shown in Table 9. AC and GBM have shown slightly better performance with a 0.62 value of accuracy which is not good enough. Results revealed that the FastText subword did not show good results in combination with machine learning models in classifying deep fake text.

TABLE 9. Classification result using FastText Subword.

Model	Accuracy	Precision	Recall	F1-Score
DT	0.56	0.56	0.56	0.56
LR	0.53	0.53	0.53	0.52
AC	0.62	0.62	0.62	0.62
SGC	0.55	0.55	0.55	0.54
RF	0.61	0.61	0.61	0.61
GBM	0.62	0.63	0.62	0.61
ETC	0.59	0.59	0.59	0.59
NB	0.56	0.60	0.56	0.52

C. RESULTS OF DEEP LEARNING MODELS

This study employs some state-of-the-art deep learning models including CNN, LSTM and CNN-LSTM. These models have been extensively used for text classification in the literature. The architecture of these deep learning models is presented in Table 4.

Embedding layers, Dropout layers, and dense layers are used by all deep learning models. The embedding layer accepts input and turns each word in reviews into vector form for model training. The dropout layer is used to lower the likelihood of model over-fitting and reduce the complexity of model learning by randomly removing neurons. To produce the required output, the dense layer is combined with the neurons and a Softmax activation algorithm. Models are constructed with a categorical cross-entropy function, and the 'Adam' optimizer is utilised to optimise parameters.

Experimental results of deep learning models using FastText and FastText Subword embeddings are presented in Table 10 and Table 11 respectively. Results revealed that deep learning models have shown better performance with FastText as compared to the FastText subword. CNN-LSTM has shown the lowest results using the FastText subword with a 0.67 value of accuracy. LSTM using FastText has shown a 0.71 accuracy score, 0.82 precision, 0.78 recall and 0.80 F1 score for deepfake text detection. CNN-LSTM has attained a 0.69 value of accuracy, 0.78 precision, 0.75 recall and 0.76 F1 Score. On the other hand, CNN in combination with FastText has shown superior performance among all other combinations of deep learning and machine learning models with a 0.93 value of accuracy, 0.92 value of precision, 0.95 recall and 0.93 F1 Score.

TABLE 10. Classification result using deep learning models using FastText.

Model	Accuracy	Precision	Recall	F1-Score
CNN	0.93	0.92	0.95	0.93
LSTM	0.71	0.82	0.78	0.80
CNN-LSTM	0.69	0.78	0.75	0.76

TABLE 11. Classification result using deep learning models using FastText Subword.

Model	Accuracy	Precision	Recall	F1-Score
CNN	0.87	0.84	0.81	0.83
LSTM	0.68	0.73	0.73	0.73
CNN-LSTM	0.67	0.71	0.74	0.75

D. DISCUSSION

Performance comparisons of classifiers utilizing various feature representation approaches were performed on the dataset including deepfake tweets. The effect of TF, TF-IDF, FastText, and FastText Subword on tweets has been examined in order to discover bot-generated tweets. Separate comparisons of accuracy, precision, recall, and F1 score have been provided. Figure 5 presents the training and testing accuracy, precision, and recall of the proposed model.

Figure 6 presents the performance comparison of deep learning models using FastText and FastText Subwords. It can be observed that LSTM and CNN-LSTM have shown better performance with FastText. However, CNN using FastText has shown the highest accuracy score with a 0.93 value. In the precision comparison of models, all three models show improved performance with FastText. While CNN using Fast Text has achieved the highest precision score among all classifiers.

Figure 6 also illustrates the comparison of deep learning models in terms of recall. The lowest recall values have been achieved by CNN-LSTM using the FastText subword. CNN using FastText surpassed other models with a 0.93 value of F1 Score.

According to the explanation above, the classifiers get the best results for deepfake text classification when trained using FastText word embedding. This study considers the use of Word embedding techniques due to the efficiency of FastText embeddings in various text classification tasks [74], [75]. By employing word embeddings, textual input is effectively transformed into numerical vectors that encapsulate meaningful semantic relationships between words. Consequently, the machine learning model can learn and identify

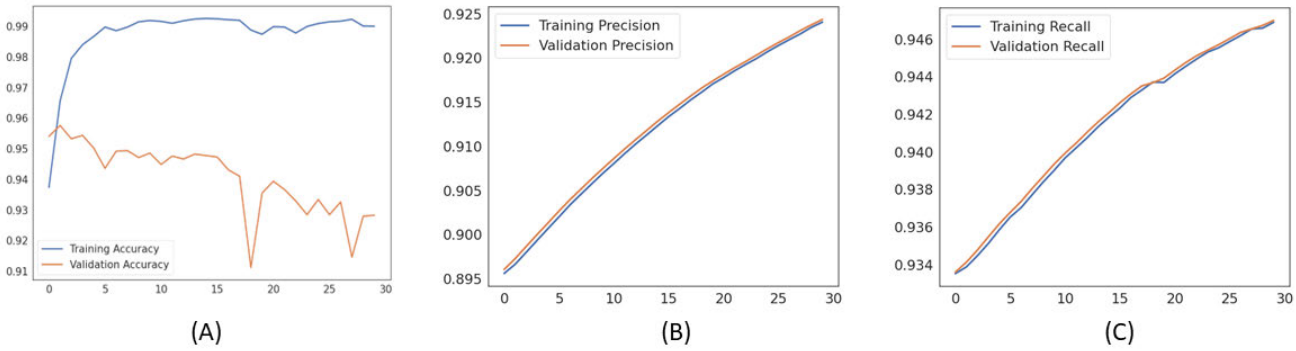

FIGURE 5. Training and testing results of CNN model. A) Accuracy Curve, B) Precision Curve, C) Recall Curve.

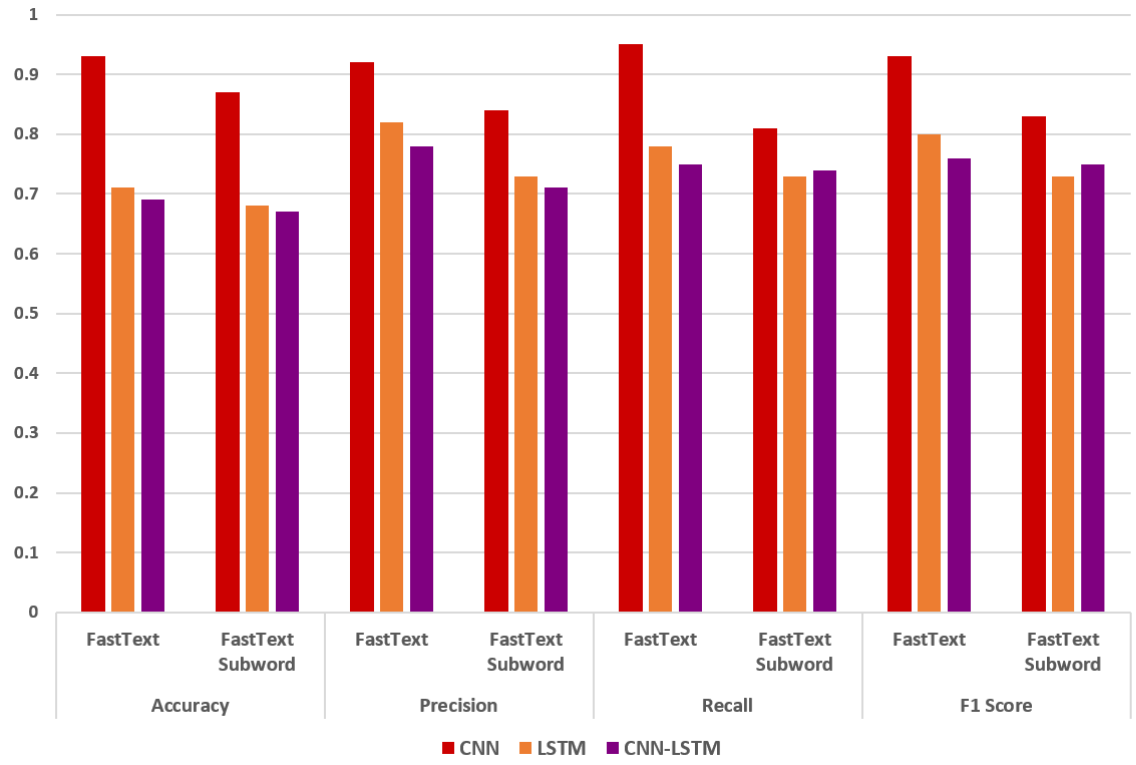

FIGURE 6. Performance comparison of deep learning models.

intricate patterns and associations between words specific to the given task. Machine learning models have shown good results using TF-IDF. TF-IDF captures the word's importance while FastText represents words as continuous vectors in a high-dimensional space to capture semantic and grammatical information. It takes subword information into consideration, allowing it to handle out-of-vocabulary terms and catch morphological similarities. Overall, CNN utilizing FastText achieved the best performance across all assessment measures. By minimising bias and variance, the randomization and optimisation features make CNN more efficient in text classification.

1) PERFORMANCE COMPARISON WITH PREVIOUS STATE-OF-THE-ART

The performance of the proposed model is evaluated by comparing it with state-of-the-art studies in the field. In order to ensure the most up-to-date results, a recently published work [19] on the same dataset has been selected for comparison that has achieved the best results with two powerful transfer learning models namely RoBERTa (Robustly Optimized BERT approach) and BERT (Bidirectional Encoder Representations from Transformers). Notably, Roberta, when applied to the same dataset, achieved an accuracy of 0.89 and an F1 score of 0.89. Similarly, BERT, when applied to the same

dataset, also reported an accuracy of 0.89 and an F1 score of 0.89.

In contrast, the current study leveraged the combination of CNN features and FastText for deepfake text detection. The results of this study outperformed existing state-of-the-art approaches, yielding a classification accuracy of 0.93. This indicates that the proposed model offers a significant improvement over previous methods in accurately identifying deepfake text. The approach employed in this study offers distinct advantages compared to complex transfer learning models, such as RoBERTa and BERT. The utilization of a simple CNN model structure provides several benefits. Firstly, it avoids the need for extensive training time and computational resources that are typically required for fine-tuning transfer learning models. This makes the proposed approach more accessible and efficient, especially for researchers and practitioners with limited resources.

Additionally, the fixed vocabulary size of transfer learning models can pose challenges when encountering out-of-vocabulary terms. In contrast, the CNN model used in this study does not suffer from such limitations. It can effectively handle out-of-vocabulary terms without compromising performance, as it does not rely on pre-defined vocabularies. This flexibility allows the model to better adapt to diverse and evolving textual data. By incorporating CNN features and harnessing the power of FastText, the proposed model achieves higher accuracy and demonstrates better performance compared to the selected state-of-the-art approaches. These findings highlight the effectiveness and superiority of the proposed model in tackling the challenge of deepfake text detection. Overall, the results obtained from the comparison with existing studies validate the proficiency of the proposed model and establish its competitive advantage in the field of deepfake text detection.

TABLE 12. Results comparison with state-of-the-art models from literature.

Models	Accuracy	F1-Score
ROBERTA [19]	0.896	0.89
BERT [19]	0.891	0.89
Proposed Approach	0.93	0.93

VI. CONCLUSION AND FUTURE WORK

Deepfake text detection is a critical and challenging task in the era of misinformation and manipulated content. This study aimed to address this challenge by proposing an approach for deepfake text detection and evaluating its effectiveness. A dataset containing tweets of bots and humans is used for analysis by applying several machine learning and deep learning models along with feature engineering techniques. Well-known feature extraction techniques: Tf and TF-IDF and word embedding techniques: FastText and FastText subwords are used. By leveraging a combination of techniques such as CNN and FastText, the proposed approach demonstrated promising results with a 0.93 accuracy score in accurately identifying deepfake text. Furthermore, the results of the proposed approach are compared with other state-of-the-art transfer learning models from previous literature.

Overall, the adoption of a CNN model structure in this study shows its superiority in terms of simplicity, computational efficiency, and handling out-of-vocabulary terms. These advantages make the proposed approach a compelling option for text detection tasks, demonstrating that sophisticated performance can be achieved without the need for complex and time-consuming transfer learning models. The findings of this study contribute to advancing the field of deepfake detection and provide valuable insights for future research and practical applications.

As social media continues to play a significant role in shaping public opinion, the development of robust deepfake text detection techniques is imperative to safeguard genuine information and preserve the integrity of democratic processes. The challenges and opportunities in the emerging trend of quantum machine learning are highlighted in [76] and the quantum approach to detect deepfake text is presented in [77]. In future research, the quantum NLP and other cutting-edge methodologies will be applied for more sophisticated and efficient detection systems, to fight against the spread of misinformation and deceptive content on social media platforms.

REFERENCES

[1] J. P. Verma and S. Agrawal, "Big data analytics: Challenges and applications for text, audio, video, and social media data," *Int. J. Soft Comput., Artif. Intell. Appl.*, vol. 5, no. 1, pp. 41–51, Feb. 2016.

[2] H. Siddiqui, E. Healy, and A. Olmsted, "Bot or not," in *Proc. 12th Int. Conf. Internet Technol. Secured Trans. (ICITST)*, Dec. 2017, pp. 462–463.

[3] M. Westerlund, "The emergence of deepfake technology: A review," *Technol. Innov. Manage. Rev.*, vol. 9, no. 11, pp. 39–52, Jan. 2019.

[4] J. Ternovski, J. Kalla, and P. M. Aronow, "Deepfake warnings for political videos increase disbelief but do not improve discernment: Evidence from two experiments," Ph.D. dissertation, Dept. Political Sci., Yale Univ., 2021.

[5] S. Vosoughi, D. Roy, and S. Aral, "The spread of true and false news online," *Science*, vol. 359, no. 6380, pp. 1146–1151, Mar. 2018.

[6] S. Bradshaw, H. Bailey, and P. N. Howard, "Industrialized disinformation: 2020 global inventory of organized social media manipulation," Comput. Propaganda Project Oxford Internet Inst., Univ. Oxford, Oxford, U.K., Tech. Rep., 2021.

[7] C. Grimme, M. Preuss, L. Adam, and H. Trautmann, "Social bots: Human-like by means of human control?" *Big Data*, vol. 5, no. 4, pp. 279–293, Dec. 2017.

[8] X. Liu, Y. Zheng, Z. Du, M. Ding, Y. Qian, Z. Yang, and J. Tang, "GPT understands, too," 2021, *arXiv:2103.10385*.

[9] R. Zellers, A. Holtzman, H. Rashkin, Y. Bisk, A. Farhadi, F. Roesner, and Y. Choi, "Defending against neural fake news," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst. (NIPS)*, Dec. 2019, pp. 9054–9065, Art. no. 812.

[10] L. Beckman, "The inconsistent application of internet regulations and suggestions for the future," *Nova Law Rev.*, vol. 46, no. 2, p. 277, 2021, Art. no. 2.

[11] J.-S. Lee and J. Hsiang, "Patent claim generation by fine-tuning OpenAI GPT-2," *World Pat. Inf.*, vol. 62, Sep. 2020, Art. no. 101983.

[12] R. Dale, "GPT-3: What's it good for?" *Natural Lang. Eng.*, vol. 27, no. 1, pp. 113–118, 2021.

[13] W. D. Heaven, "A GPT-3 bot posted comments on Reddit for a week and no one noticed," MIT Technol. Rev., Cambridge, MA, USA, Tech. Rep., Nov. 2020, p. 2020, vol. 24. [Online]. Available: www.technologyreview.com

[14] S. Gehrmann, H. Strobelt, and A. M. Rush, "GLTR: Statistical detection and visualization of generated text," 2019, *arXiv:1906.04043*.

[15] D. I. Adelani, H. Mai, F. Fang, H. H. Nguyen, J. Yamagishi, and I. Echizen, "Generating sentiment-preserving fake online reviews using neural language models and their human- and machine-based detection," in *Proc. 34th Int. Conf. Adv. Inf. Netw. Appl. (AINA)*. Cham, Switzerland: Springer, 2020, pp. 1341–1354.
[16] R. Zellers, A. Holtzman, H. Rashkin, Y. Bisk, A. Farhadi, F. Roesner, and Y. Choi, "Grover—A state-of-the-art defense against neural fake news," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32. Curran Associates, 2019. [Online]. Available: http://papers.nips.cc/paper/9106-defending-against-neural-fake-news.pdf
[17] N. S. Keskar, B. McCann, L. R. Varshney, C. Xiong, and R. Socher, "CTRL: A conditional transformer language model for controllable generation," 2019, *arXiv:1909.05858*.
[18] A. Uchendu, Z. Ma, T. Le, R. Zhang, and D. Lee, "TURINGBENCH: A benchmark environment for Turing test in the age of neural text generation," 2021, *arXiv:2109.13296*.
[19] T. Fagni, F. Falchi, M. Gambini, A. Martella, and M. Tesconi, "TweepFake: About detecting deepfake tweets," *PLoS ONE*, vol. 16, no. 5, May 2021, Art. no. e0251415.
[20] H. Stiff and F. Johansson, "Detecting computer-generated disinformation," *Int. J. Data Sci. Anal.*, vol. 13, no. 4, pp. 363–383, May 2022.
[21] M. Gambini, T. Fagni, F. Falchi, and M. Tesconi, "On pushing deepfake tweet detection capabilities to the limits," in *Proc. 14th ACM Web Sci. Conf.*, Jun. 2022, pp. 154–163.
[22] R. Tolosana, R. Vera-Rodriguez, J. Fierrez, A. Morales, and J. Ortega-Garcia, "Deepfakes and beyond: A survey of face manipulation and fake detection," *Inf. Fusion*, vol. 64, pp. 131–148, Dec. 2020.
[23] T. T. Nguyen, Q. V. H. Nguyen, D. T. Nguyen, D. T. Nguyen, T. Huynh-The, S. Nahavandi, T. T. Nguyen, Q.-V. Pham, and C. M. Nguyen, "Deep learning for deepfakes creation and detection: A survey," 2019, *arXiv:1909.11573*.
[24] T. Chen, A. Kumar, P. Nagarsheth, G. Sivaraman, and E. Khoury, "Generalization of audio deepfake detection," in *Proc. Speaker Lang. Recognit. Workshop (Odyssey)*, Nov. 2020, pp. 132–137.
[25] M. Wolff and S. Wolff, "Attacking neural text detectors," 2020, *arXiv:2002.11768*.
[26] J. Pu, Z. Sarwar, S. M. Abdullah, A. Rehman, Y. Kim, P. Bhattacharya, M. Javed, and B. Viswanath, "Deepfake text detection: Limitations and opportunities," 2022, *arXiv:2210.09421*.
[27] F. Kateb and J. Kalita, "Classifying short text in social media: Twitter as case study," *Int. J. Comput. Appl.*, vol. 111, no. 9, pp. 1–12, Feb. 2015.
[28] A. Garcia-Silva, C. Berrio, and J. M. Gómez-Pérez, "An empirical study on pre-trained embeddings and language models for bot detection," in *Proc. 4th Workshop Represent. Learn. NLP (RepL4NLP)*, 2019, pp. 148–155.
[29] J. Lundberg, J. Nordqvist, and A. Matosevic, "On-the-fly detection of autogenerated tweets," 2018, *arXiv:1802.01197*.
[30] R. Gorwa and D. Guilbeault, "Unpacking the social media bot: A typology to guide research and policy," *Policy Internet*, vol. 12, no. 2, pp. 225–248, Jun. 2020.
[31] S. Suwajanakorn, S. M. Seitz, and I. Kemelmacher-Shlizerman, "Synthesizing obama: Learning lip sync from audio," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 1–13, Aug. 2017.
[32] J. Thies, M. Zollhöfer, M. Stamminger, C. Theobalt, and M. Nießner, "Face2Face: Real-time face capture and reenactment of RGB videos," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2387–2395.
[33] C. Chan, S. Ginosar, T. Zhou, and A. Efros, "Everybody dance now," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 5932–5941.
[34] Y. Jia, Y. Zhang, R. Weiss, Q. Wang, J. Shen, F. Ren, P. Nguyen, R. Pang, I. L. Moreno, and Y. Wu, "Transfer learning from speaker verification to multispeaker text-to-speech synthesis," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 4485–4495.
[35] Y. Wang, K. Wang, Y. Wang, D. Guo, H. Liu, and F. Sun, "Audio-visual grounding referring expression for robotic manipulation," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2022, pp. 9258–9264.
[36] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, p. 9, 2019.
[37] Y. Zhou, J. Yang, D. Li, J. Saito, D. Aneja, and E. Kalogerakis, "Audio-driven neural gesture reenactment with video motion graphs," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 3408–3418.
[38] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," OpenAI, Amer. AI Res. Lab., Tech. Rep., 2018.
[39] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.
[40] C. Li, X. Gao, Y. Li, B. Peng, X. Li, Y. Zhang, and J. Gao, "Optimus: Organizing sentences via pre-trained modeling of a latent space," 2020, *arXiv:2004.04092*.
[41] I. Solaiman, M. Brundage, J. Clark, A. Askell, A. Herbert-Voss, J. Wu, A. Radford, G. Krueger, J. W. Kim, S. Kreps, M. McCain, A. Newhouse, J. Blazakis, K. McGuffie, and J. Wang, "Release strategies and the social impacts of language models," 2019, *arXiv:1908.09203*.
[42] P. von Platen, "How to generate text: Using different decoding methods for language generation with transformers," Hugging Face, Manhattan, NY, USA, Tech. Rep., 2020.
[43] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "XLNet: Generalized autoregressive pretraining for language understanding," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.
[44] A. Bakhtin, S. Gross, M. Ott, Y. Deng, M. Ranzato, and A. Szlam, "Real or fake? Learning to discriminate machine from human generated text," 2019, *arXiv:1906.03351*.
[45] S. Feng, "TwiBot-22: Towards graph-based Twitter bot detection," 2022, *arXiv:2206.04564*.
[46] S. H. Moghaddam and M. Abbaspour, "Friendship preference: Scalable and robust category of features for social bot detection," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 2, pp. 1516–1528, Mar. 2023.
[47] N. Hajli, U. Saeed, M. Tajvidi, and F. Shirazi, "Social bots and the spread of disinformation in social media: The challenges of artificial intelligence," *Brit. J. Manage.*, vol. 33, no. 3, pp. 1238–1253, Jul. 2022.
[48] Y. Mirsky and W. Lee, "The creation and detection of deepfakes: A survey," *ACM Comput. Surv.*, vol. 54, no. 1, pp. 1–41, 2021.
[49] D. Dukić, D. Keča, and D. Stipić, "Are you human? Detecting bots on Twitter using BERT," in *Proc. IEEE 7th Int. Conf. Data Sci. Adv. Anal. (DSAA)*, Oct. 2020, pp. 631–636.
[50] S. Kudugunta and E. Ferrara, "Deep neural networks for bot detection," *Inf. Sci.*, vol. 467, pp. 312–322, Oct. 2018.
[51] M. Heidari and J. H. Jones Jr., "BERT model for social media bot detection," Mason Archival Repository Service, George Mason Univ. Libraries, Tech. Rep., 2022.
[52] K. E. Daouadi, R. Z. Rebaï, and I. Amous, "Bot detection on online social networks using deep forest," in *Proc. 8th Comput. Sci. On-Line Conf.* Cham, Switzerland: Springer, 2019, vol. 2, no. 8, pp. 307–315.
[53] D. Ippolito, D. Duckworth, C. Callison-Burch, and D. Eck, "Automatic detection of generated text is easiest when humans are fooled," 2019, *arXiv:1911.00650*.
[54] J. Zhao, X. Liu, Q. Yan, B. Li, M. Shao, and H. Peng, "Multi-attributed heterogeneous graph convolutional network for bot detection," *Inf. Sci.*, vol. 537, pp. 380–393, Oct. 2020.
[55] B. Wu, L. Liu, Y. Yang, K. Zheng, and X. Wang, "Using improved conditional generative adversarial networks to detect social bots on Twitter," *IEEE Access*, vol. 8, pp. 36664–36680, 2020.
[56] S. Najari, M. Salehi, and R. Farahbakhsh, "GANBOT: A GAN-based framework for social bot detection," *Social Netw. Anal. Mining*, vol. 12, no. 1, pp. 1–11, Dec. 2022.
[57] B. Guo, X. Zhang, Z. Wang, M. Jiang, J. Nie, Y. Ding, J. Yue, and Y. Wu, "How close is ChatGPT to human experts? Comparison corpus, evaluation, and detection," 2023, *arXiv:2301.07597*.
[58] R. Shijaku and E. Canhasi, "ChatGPT generated text detection," Tech. Rep., 2023.
[59] S. Mitrović, D. Andreoletti, and O. Ayoub, "ChatGPT or human? Detect and explain. Explaining decisions of machine learning model for detecting short ChatGPT-generated text," 2023, *arXiv:2301.13852*.
[60] Y. Ma, J. Liu, F. Yi, Q. Cheng, Y. Huang, W. Lu, and X. Liu, "AI vs. human—Differentiation analysis of scientific content generation," 2023, *arXiv:2301.10416*.
[61] B. Yu, "An evaluation of text classification methods for literary study," *Literary Linguistic Comput.*, vol. 23, no. 3, pp. 327–343, Sep. 2008.
[62] S. Robertson, "Understanding inverse document frequency: On theoretical arguments for IDF," *J. Document.*, vol. 60, no. 5, pp. 503–520, Oct. 2004.
[63] C. Qiao, B. Huang, G. Niu, D. Li, D. Dong, W. He, D. Yu, and H. Wu, "A new method of region embedding for text classification," in *Proc. ICLR*, 2018, pp. 135–146.
[64] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Trans. Assoc. Comput. Linguistics*, vol. 5, pp. 135–146, Dec. 2017.

[65] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
[66] S. B. Kotsiantis, "Decision trees: A recent overview," *Artif. Intell. Rev.*, vol. 39, no. 4, pp. 261–283, Apr. 2013.
[67] D. G. Kleinbaum, K. Dietz, M. Gail, M. Klein, and M. Klein, *Logistic Regression*. New York, NY, USA: Springer-Verlag, 2002.
[68] N. Ketkar and N. Ketkar, "Stochastic gradient descent," in *Deep Learning With Python: A Hands-On Introduction*. Berkeley, CA, USA: Apress, 2017, pp. 113–132.
[69] A. R. Javed, Z. Jalil, S. A. Moqurrab, S. Abbas, and X. Liu, "Ensemble AdaBoost classifier for accurate and fast detection of botnet attacks in connected vehicles," *Trans. Emerg. Telecommun. Technol.*, vol. 33, no. 10, p. e4088, 2022.
[70] A. Natekin and A. Knoll, "Gradient boosting machines, a tutorial," *Frontiers Neurorobotics*, vol. 7, p. 21, Dec. 2013.
[71] G. I. Webb, E. Keogh, and R. Miikkulainen, "Naïve Bayes," in *Encyclopedia of Machine Learning*, vol. 15, C. Sammut and G. I. Webb, Eds. USA: Springer, 2017, pp. 713–714.
[72] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Mach. Learn.*, vol. 63, no. 1, pp. 3–42, Apr. 2006.
[73] H. Xie, L. Zhang, and C. P. Lim, "Evolving CNN-LSTM models for time series prediction using enhanced grey wolf optimizer," *IEEE Access*, vol. 8, pp. 161519–161541, 2020.
[74] S. S. Birunda and R. K. Devi, "A review on word embedding techniques for text classification," in *Innovative Data Communication Technologies and Application*. Singapore: Springer, 2021, pp. 267–281.
[75] E. Batbaatar, M. Li, and K. H. Ryu, "Semantic-emotion neural network for emotion recognition from text," *IEEE Access*, vol. 7, pp. 111866–111878, 2019.
[76] R. Guarasci, G. De Pietro, and M. Esposito, "Quantum natural language processing: Challenges and opportunities," *Appl. Sci.*, vol. 12, no. 11, p. 5651, Jun. 2022.
[77] B. Mishra and A. Samanta, "Quantum transfer learning approach for deepfake detection," *Sparklinglight Trans. Artif. Intell. Quantum Comput.*, vol. 2, no. 1, pp. 17–27, 2022.

SAIMA SADIQ received the Ph.D. degree in computer science from the Khwaja Fareed University of Engineering and IT (KFUEIT), in September 2020 Her current research interests include data mining, machine learning, and deep learning-based text mining.

TURKI ALJREES received the Ph.D. degree in computer science from Glasgow University, Scotland. He is currently an Assistant Professor with the Computer Engineering Department, University of Hafr Al Batin, Saudi Arabia. His current research interests include computer vision, optimization techniques, and performance enhancement.

SALEEM ULLAH was born in Ahmedpur East, Pakistan, in 1983. He received the B.Sc. degree in computer science from Islamia University Bahawalpur, Pakistan, in 2003, the master's degree in computer science from Bahauddin Zakariya University, Multan, in 2005, and the Ph.D. degree from Chongqing University, China, in 2012. From 2006 to 2009, he was a network/IT administrator in different companies. From August 2012 to February 2016, he was an Assistant Professor with Islamia University Bahawalpur. He has been an Associate Dean with the Khwaja Fareed University of Engineering and Information Technology, Rahim Yar Khan, since February 2016. He has almost 14 years of industry experience in the field of IT. He is an active researcher in the field of adhoc networks, the IoTs, congestion control, data science, and network security.