Digital Object Identifier

A Framework for Hate Speech Detection using Deep Convolutional Neural Network

PRADEEP KUMAR ROY[1,3], ASIS KUMAR TRIPATHY (MEMBER, IEEE) [1], TAPAN KUMAR DAS (MEMBER, IEEE)[1], AND XIAO-ZHI GAO[2]
[1]School of Information Technology, Vellore Institute of Technology, Vellore
e-mail: pkroynitp@gmail.com, asistripathy@gmail.com, tapandas05@gmail.com
[2]School of Computing, University of Eastern Finland, Kuopio, Finland
e-mail:xiao.z.gao@gmail.com
[3]Current Institute: Indian Institute of Information Technology, Surat, Gujarat

Corresponding author: Pradeep Kumar Roy (e-mail: pkroynitp@gmail.com).

ABSTRACT The rapid growth of Internet users led to unwanted cyber issues, including cyberbullying, hate speech, and many more. This paper deals with the problems of hate speech on Twitter. Hate speech appears to be an inflammatory kind of interaction process that uses misconceptions to express a hate ideology. The hate speech focuses on various protected aspects, including gender, religion, race, and disability. Owing to hate speech, sometimes unwanted crimes are going to happen as someone or a group of people get disheartened. Hence, it is essential to monitor user's posts and filter the hate speech related post before it is spread. However, Twitter receives more than six hundred tweets per second and about 500 million tweets per day. Manually filtering any information from such a huge incoming traffic is almost impossible. Concerning to this aspect, an automated system is developed using the Deep Convolutional Neural Network (DCNN). The proposed DCNN model utilises the tweet text with GloVe embedding vector to capture the tweets' semantics with the help of convolution operation and achieved the precision, recall and F1-score value as 0.97, 0.88, 0.92 respectively for the best case and outperformed the existing models.

INDEX TERMS Convolutional Neural Network, Hate Speech, LSTM, Tf-Idf, Twitter

I. INTRODUCTION

A large number of people have recently joined Online Social Networking (OSN) websites, mainly due to worldwide availability of low-cost Internet. According to a survey conducted by *Statista*, by January 2020, more than 4.54 billion people were active Internet users, accounting for 59% of the global population [1]. Among these users, 3.8 billion people were active users of OSN websites, which represents 83.70% of the total Internet users.

OSN websites like Facebook, WhatsApp, Wechat, Twitter, and Instagram are easily accessible and have become a favorite platform for the users to interact with each other [1]. One of the reasons behind the popularity of these platforms is the availability of information in a variety of formats such as audio, video, and images. These information ranges from politics to technology, science to music, nature to space. There is hardly any walk of life, which is untouched by these OSNs, they meet the needs of every individual and therefore, people prefer to spend their time on OSN [2], [3].

Among all OSNs, Twitter is a special type of OSN where the length of each message is limited to 280 characters. Also, instead of sending a request to become a friend, Twitter has the follower and followee based interface. That means, if you want to see updates for other users (friends, relatives, favorite actors), you have to follow them. Similarly, if someone wants to see your posts, they have to follow you through [4].

Generally, people are using Twitter to get updates from personal and professional connections, as well as to see the current news of the world. Due to no restriction on what to post, people feel free to post any post on Twitter; as a result, it is effortless to post negative comments, hateful messages on Twitter [5]–[7].

Most recently, Facebook and Twitter received a large number of Hate Speech (*HS*) related posts on Delhi's (India capital) Shaheen bag protest against *National Register*

[1]https://www.statista.com/statistics/617136/digital-population-worldwide/[access: 31-03-2020]

of Citizens (NRC), *Citizenship Amendment Act* (CAA) and *National Population Register* (NPR), which began on 11 December, 2019[2]. Again during the COVID-19 coronavirus epidemic, *HS*-related tweets were started trending on Twitter (with the hashtag `#hatespeech, #HateSpeech`)[3] and Facebook.

The above-mentioned issue has attracted researchers over the past few years and has therefore proposed models using machine learning and deep learning techniques [8]–[14]. However, existing models do not meet the required needs, as many *HS* related tweets are still available on Twitter and floating across the network. This prompted us to develop a model that would capture the maximum number of *HS* related posts. The CNN model has been successfully used by current researchers to address various issues related to the text domain, including sentiment analysis, question answering, document classification, sentence classification [15]–[17], spam filtering and others [18]–[20]. By following them, this research also uses a deep convolution neural network (DCNN) to address the hate speech detection issue. DCNN is capable of capturing the semantics of the sentence by performing the convolution operations over the tweets. We also tested other deep neural network-based models such as Long Short-Term Memory (LSTM), and Convolutional-LSTM (C-LSTM) network for the same and found the DCNN model is a better choice for this research. The main contributions are as follows:

- We proposed a deep convolutional neural network (DCNN) framework to improve the hate speech post detection on Twitter.
- The proposed DCNN model requires only tweet texts as input for prediction; hence it reduces the overhead of the manual feature extraction process.
- The proposed system achieved good prediction accuracy on an imbalanced dataset and outperformed existing models.

The rest of the article is organized as follows: Section II highlights the existing research on hate speech. Section III describe the problem statement. In section IV, the proposed model is explained. The experimental results is discussed in Section V. Finally Section VI concludes this work with limitations and future scope of research.

II. RELATED WORKS

There is no proper or clear definition of *HS* is described [21]. The statement that hurt someone may be called as *HS* [8]. Sometimes, the *HS* may also be called abusive statement [22], however, few works used the term hate speech in their research [23]–[28]. In order to detect the *HS* on social media platform, researchers mainly used two methodologies: (i) traditional machine learning approach and (ii) deep learning based approach. The subsequent subsections discuss the existing researches using these two approaches.

[2]https://www.huffingtonpost.in/[accessed on 27/05/2020]
[3]https://newscentral24x7.com/ [accessed on 27/05/2020]

A. HATE SPEECH DETECTION WITH MACHINE LEARNING APPROACH

Warner and Hirschberg [26] collected the data for their research from two websites (i) Yahoo! and (ii) American Jewish Congress. Using the SVM^{light} classifier [29] they achieved precision, recall, F1-score and accuracy value of 0.68, 0.60, 0.64, 95% respectively for the best case. Kwok and Wang [25] used Bag-of-Words feature extraction technique with Naive Bayes classifier to classify the tweets. The model achieved 76% of accuracy on 10-fold cross validation setting for the best case. They said that the BOW model is not sufficient to classify the *HS* related tweets. They only used uni-gram feature to achieve this accuracy and said that, if bi-gram and tweet's sentiment score will be added in feature set, then it may provide better accuracy.

Burnap and Williams [27] collected 450,000 tweets for their research. *N*-grams (1-5) word features were extracted from tweets and the supervised model was used to classify them. Three classifiers (i) Bayesian logistic regression, (ii) Support Vector Machine (SVM) and (iii) voted ensemble classifier were tested. The best results were obtained using the Voted Ensemble Classifier where the accuracy, recall and F1-scores were 0.89, 0.69, 0.77, respectively. Waseem and Hovy [30] provided 16k labelled dataset for *HS* detection. They extracted uni-, bi-, tri-, and quad-gram features from tweets. Using the logistic regression classifier on 10-fold cross validation setting obtained the F1-score of 73.89%, 73.66%, 73.62%, 73.47% with *Gender*, *Length*, $\{Gender + location\}$ and $\{Gender + location + length\}$ based feature set respectively.

Davidson et al. [8] proposed an automated model to detect the *HS*. They crawled the data from crowd source and labelled into three categories: (i) *HS*, (ii) Offensive and (iii) Neither. From the labelled dataset, they extracted uni-, bi-, tri- and quad-gram of features weighted by the *tf-idf* technique. To reduce the dimension of the dataset, logistic regression with *L1* regularization technique was used and then multiple classifiers such as decision tree, random forest, linear SVM, naive bayes were used to test the performance with 5-fold cross validation setting. Finally, by following the previous research [9], [27], they also used logistic regression with *L2* regularization technique to classify the tweets and achieved the F1-score of 0.90. However, for *HS*, the miss-classification rate was 40%. Gao and Huang [10] proposed a context aware based *HS* detection model. They used logistic regression and LSTM model and found both the models surpassed from the baseline model (Char based) by 3% and 4%.

B. HATE SPEECH DETECTION WITH DEEP LEARNING APPROACH

Djuric et al. [28] developed a model for identification of *HS* in user comments. They used continuous bag of words (CBOW) and paragraph2vec to represent comments in a low-dimensional space. These features have been fed into a binary classifier to classify the comments into hateful or clean comments and have achieved the best AUC value of 0.80 with paragraph2vec. Park and Fung [31] used logistic regression and CNN model to classify tweets. They used traditional machine learning classifiers and deep learning models for their work and observed that the combination of two models performed better than the individual models.

Zhang et al. [14] suggested a combination of CNN and Gated Recurrent Unit network based model for *HS* detection. They experimented with seven publicly available datasets and recorded that their model outperformed six of the seven models by 1 to 14% of the higher F1-score average. The model capable of capturing both semantic and sequence-based features in short text. Kamble and Joshi [32] worked on English-Hindi mixed tweets for *HS* detection. They developed their embeddings with the large corpus of code-mixed data to build the model. The experimental results confirmed that the developed code-mixed embedding provided better performance compared to the pre-trained word embedding. Several classifier models such as SVM, Random Forest, CNN-1D, LSTM, and Bi-LSTM models were used for the experiment. The best performance was obtained with CNN-1D model, where the precision, recall, and F1-score value was 83.34, 78.51, 80.85 respectively.

There were many models developed by the researchers to address the *HS* issues on Twitter using traditional machine learning and deep learning based models. They used the features extracted using the Bag-of-Words [25], [28], n-gram [27], [30], tf-idf [8], one-hot encoding technique. The traditional machine learning based models required strong feature engineering to predict the *HS* tweets accurately, which is a tedious task and a separate research domain. Few researchers used perceptron model to predict the *HS*; the perceptron model worked with one-hot encoded input and hence did not preserve the semantic of the tweets, which leads to a high rate of misclassification. This work uses a convolutional neural network-based model to overcome these limitations and improve prediction accuracy.

III. PROBLEM STATEMENT

The aim of this research is to identify the maximum number of *HS* related tweets from Twitter as soon as it is posted by users. This issue falls within the context of a binary classification problem where tweets are classified into two classes, i.e. either Hate Speech (*HS*) or Non-Hate Speech (*NHS*). Mathematically, the problem statement can be represented as follows: suppose we have a large number of incoming tweets, $\{t_1, t_2, t_3, ...t_n\}$, which may fall under the $\{c_1, c_2, c_3, ..., c_n\}$ categories. Then the $\{t_1, t_2, t_3, ...t_n\}$ tweets will be labelled in two classes, i.e. $c_1$and c_2, where c_1 represents *HS* related tweets and c_2 represents other tweets (*NHS*). For the baseline model, traditional machine learning based classifiers were used and compares performance with the proposed DCNN deep neural model. The required features for baseline models was extracted using *tf-idf* techniques. In contrast, the proposed DCNN model (Figure 1) was itself extracted the essential features from the tweet text using the convolution process with the help of multiple filters present in intermediate layers. The complete working of the proposed DCNN model explained using Algorithm 1 and Algorithm 2.

IV. EXPERIMENTAL STUDY

We used machine learning models as a baseline and compared it with the proposed DCNN model and LSTM network. Machine learning based models requires a feature vector as input, which was extracted using the *tf-idf* technique. For deep learning model, multiple layers of convolution with different sizes of filters are used to extract the hidden contextual features from the tweet text. The dataset used for this study is taken from *Kaggle.com*[4]. It was prepared by collecting the tweets from twitter. The dataset description was missing on the uploaded webpage however, by manual inspection during the research we found that the dataset contained English written tweets only. The other languages tweets were not considered for this case. The developed dataset contained a total of 31,962 English written tweets, of which 29,720 tweets (92.98%) are Non-Hate Speech (*NHS*) and the remaining 2,242 tweets (7.02%) are Hate Speech (*HS*) related tweets. The data set is divided into training and test component at a ratio of 3:1 for the initial experiment, and, finally, a 10-fold cross-validation technique is used for proposed DCNN model development.

A. PROPOSED MODEL: DEEP CONVOLUTIONAL NEURAL NETWORK

The *tf-idf* feature extraction technique is effectively used to solve many text classification problem. However, it has several limitations, including (i) Document similarity is measured directly in word-count space, which may be sluggish for colossal vocabulary. (ii) Claims that the separate word counts have independent proof of similarity. (iii) It does not exploit semantic similarities between words.

The advantages of the feature extraction using the convolution process of CNN model over *tf-idf* technique are as follows: it uses multiple hidden layer based trainable neural network architecture to extract the hidden features from the tweets [33]–[40]. The self-extraction of contextual features is independent of hand-crafted feature engineering. CNN also catches a semantically equivalent term with the help of word embeddings. As shown in Figure 1, a standard CNN model for text classification consists of many different steps, which are explained in subsequent subsections.

1) Embedding Layer

A pre-trained embedding vector GloVe [41] is used to convert the tweet text into a numerical vector shape. The word embedding maps the word of the tweet and produces a tweet matrix. Previously, as a pre-requisite of the proposed DCNN model, the number of words in each tweet makes the use of

[4]https://www.kaggle.com/vkrahul/twitter-hate-speech?select=train_E6oV3lV.csv

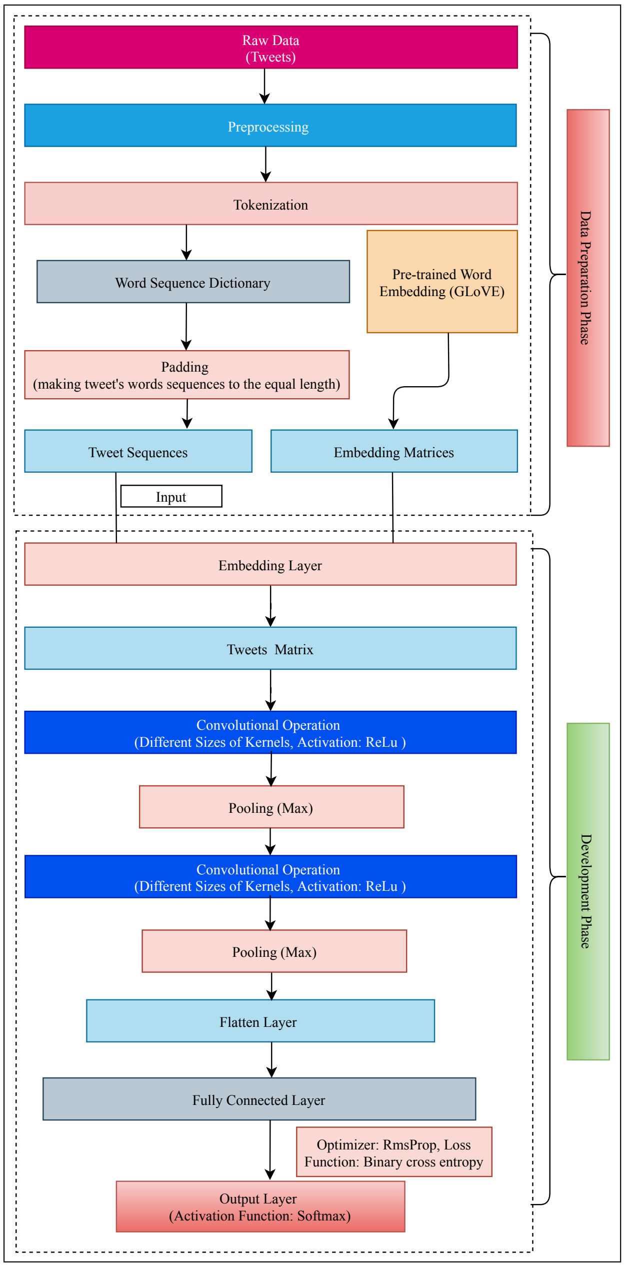

Raw Data (Tweets)
Preprocessing
Tokenization
Word Sequence Dictionary
Pre-trained Word Embedding (GLoVE)
Padding (making tweet's words sequences to the equal length)
Tweet Sequences
Embedding Matrices
Input
Data Preparation Phase
Embedding Layer
Tweets Matrix
Convolutional Operation (Different Sizes of Kernels, Activation: ReLu)
Pooling (Max)
Convolutional Operation (Different Sizes of Kernels, Activation: ReLu)
Pooling (Max)
Flatten Layer
Fully Connected Layer
Optimizer: RmsProp, Loss Function: Binary cross entropy
Output Layer (Activation Function: Softmax)
Development Phase

FIGURE 1: A Framework of the Proposed Deep Convolutional Neural Network

padding comparable. After padding, each tweet *T* has a fixed number of words that say *k*, i.e. $T = \{t_1, t_2, t_3, ...t_k\}$. Now, from the pre-trained word embedding GloVe, for all terms *k*, the respective word vector extracted and concatenated together to form a tweet matrix *T* with a size of $k \times d$, where *k* is a number of tweet words, and *d* is a pre-trained word embedding dimension.

2) Convolution Process

It helps to extract hidden features from tweets. The tweet matrix *T* obtained from the embedding layer is the input of the convolution process. In the convolution cycle, the *n*-gram filter *f* has a dimension of $n \times d$, where *n* is the number of tweet words on which the convolution is performed, and *d* is the same dimension as the *d* tweet matrix *t* has a convolution in *t*. The filter *f* only contains *n* words of *T* and stores the extracted word, again the filter *f* slides vertically to *T* and performs a convolution with the next
textitn words of *T*. The cycle will proceed until *f* is completed with the last word of *T*, i.e. t_k. This way, the *n*-gram filter *f* extracted a total of $\{(k - n) + 1\}$ number of features. If the matrix *T* has a dimension of 30×100 and filter *f* has a dimension of 5×100 then a total of 26 {((30-5)+1) = 26} characters will be extracted by the convolution method. This work uses a number of different size filters to extract contextual features from tweets.

3) Pooling

The pooling layer is used to pool out the essential features from the extracted set of features. Convolution process extracted the features *F* from the tweet text with the help of filter *f*, however, it is not necessary that all extracted features are important one. Furthermore, in order to reduce the computing overhead of the model, the selected but essential features need to be continued. The literature indicates that for the purpose of text classification, max-pooling is a better option compared to the average-pooling and min-pooling method. Hence, in order to obtain essential features, this work also uses the max-pooling approach in which maximum value is pooled from the fixed window *w*. The working of pooling operation can be understood with the following example: if the total number of features in *F* is 26, and the window size is 5 (*w*=5), then only $\lfloor \frac{F}{w} \rfloor$ (here $\lfloor \frac{26}{5} \rfloor = 5$) features will be used for further processing. This decreases model computing time significantly.

4) Fully Connected Layer

The features that have been pooled out of the pooling layer are the input of the fully connected dense layer. It analyses the pooled features and predicts the probability of tweets referring to *HS* and *NHS* tweets. The *Softmax* activation function is used for the output layer which has yielded predictions between 0 and 1, but sum of the probabilities would have to be 1. As a consequence, the highest probability will be the determining factor for the tweet classes. The working principle of the proposed model is elaborated using Algorithms 1 and 2.

Algorithm 1 describes the steps followed to prepare the data for processing. It takes the raw dataset having different attributes as input, processes it, and produces the word-vector space for each word. First, the tweets are separated and stored into a variable *text*, and their corresponding label was stored into a variable *label*. The tweets in further passes cleaned, preprocessed, and splitted into tokens using the *Tokenizer*(). Further, each token pass through the function *tokenizer.texts_to_sequences*() to assign a sequence number for future reference. The tokenized tweets again converted into key-value pair with the help of *tokenizer.word_index*

TABLE 1: Notation used in Algorithm 1 and Algorithm 2

Notation	Definition
D_f	Data with different attributes
E_m	Embedded matrix
l_e	Label converted into one hot vector form
t_p	Padding tweets to make equal in length
l_m	Maximum length of the tweet
E_d	Embedding dimension
E_m	Embedding matrix
E_v	Vector of tweet word
E_l	Embedding layer
n_n	Number of neurons
a_f	Activation function
n_f	Number of filters
F_s	Filter size
I_l	Input length
W	Network weights
e_s	Embedding sequence
e_l	Embedding layer
s_i	Sequence input
n_{ol}	Number of neurons at output layer

```
Input: $D_f$: Data with differnt attributes
Output: $E_m$[i]=Embedded Matrix
begin
    data ← $D_f$ ;
    text ← $D_f$[text];
    label ← $D_f$ [label];
    Tokenization:;
        $t_f$ ← Tokenizer();
        $t$ ← $t_f$.fit_on_texts(text);
        $t_s$ ← tokenizer.texts_to_sequences(text);
        $w_i$ ← tokenizer.word_index;
        $w_{it}$ ← len(word_index);
    Label Encoding:;
        $l_e$ ← to_categorical(label);
    Padding:;
        $t_p$ ← pad_sequences($t_s$, $l_m$);
    Preparing Embedding Matrix:;
        $E_m$ ← matrix (($w_{it}$+1) × $E_d$);
    for w, i, in word_index.items(): do
        $E_v$ ← $e_i$.get(word)
        $E_m$[i] ← $E_v$
    end
end
```

Algorithm 1: Data preparation for the proposed DCNN model

```
Input: $E_v$[i]: Vector of Word sequence
Output: Classified data
begin
    $E_l$ ← Embedding(($w_{it}$+1), $E_d$, $W = E_m$, $I_l$=$l_m$)
     ;
    convs=[];
    $F_s$={n; where n ∈ {$f_1$, $f_2$, $f_3$,.... $f_n$}};
    $s_i$ = Input(shape=($l_m$,),);
    $e_s$ = $e_l$($s_i$);
    for f in $F_s$: do
        m=Conv1D($n_f$, $f_l$=f, $a_f$=relu)($e_s$)
        m= MaxPooling1D()(m)
        convs.append(x)
    end
    m = Concatenate()(convs);
    m = Conv1D($n_f$, $F_s$, $a_f$=relu)(m);
    m = MaxPooling1D(m);
    m = Flatten()(m);
    m = Dense($n_n$, $a_f$=relu)(m);
    m = Dense($n_n$, $a_f$=relu)(m);
    p = Dense($n_{ol}$, $a_f$)(m);
    model = Model($s_i$, p);
    Compilation:;
        model.compile(loss_function,
     optimizer,metrics);
    Model Training:;
        model.fit(training
     sample,batch_size,epochs);
    Evaluation:;
        met-
     rics.classification_report(test_samples);
end
```

Algorithm 2: Proposed DCNN model

function. Now, each token has a proper key and corresponding sequence number. To fulfill the requirements, each tweet-length makes equal with the help of *pad_sequences* (). This function takes two arguments as input, first the token sequences, and second a predefined length of the sequence. The *pad_sequences* () returns the tweets having equal in size. Now, to get the word vector, a pre-trained embedding was used. For each token, a vector is extracted from the pre-trained embedding. If a particular word is not present in the pre-trained embedding corpus, it returns a default vector associated with word unknown $< unk >$. Algorithm 2 developed the embedding layers with the knowledge of the number of unique words present in the corpus (w_i), and the size of the embedding vector (E_d), the length of each tweet (l_m) and formed a tweet matrix. Different sizes of kernels (F_s) convolved over the obtained matrix and identified the features. These features flatten and pass to the dense layer to predict the predefined classes of the tweets.

B. LONG-SHORT TERM MEMORY

The proposed DCNN model achieved 0.98 F1-score for the best case, and the misclassification rate is around 2%. To mitigate this misclassification, we further used another deep neural model called LSTM. The LSTM model works well with sequential data, where the model needs to preserve the context of long-sequence [42], [43]. As it can be seen from Fig 2, a LSTM unit mainly has four gates, (i) input gate (I_t), (ii) output gate (O_t), (iii) forget gate (F_t) and (iv) memory unit (c_t). The role of (I_t) is to fetch the data into the model, the received data is processed by LSTM model and separated the useful data from the raw. Next, it is the forget gate (F_t) responsibility to throw out the irrelevant data from the cell state. Mathematically, the forget gate is defined by Eq. 1:

$$F_t = \sigma(W_f[r_{t-1},\ i_t] + B_f) \tag{1}$$

where W_f is the weight, $r_{(t-1)}$ is the output from the previous timestamp, i_t is the new input message word, and B_f is the bias. The stored data from the previous unit and the current input data is further processed in two steps as can be seen from Fig. 2. The input gate I_t and the *tanh* layer processed the data and generate c_t which added with the I_t values. the c_t and i_t are calculated by the Eq. 2, and Eq. 3:

$$c_t = tanh(W_c[h_{t-1},\ i_t] + B_c) \tag{2}$$

$$I_t = \sigma(W_I[h_{t-1},\ i_t] + B_I) \tag{3}$$

After this, the previous unit information r_{t-1} is updated to new information r_t which is defined (Eq. 4):

$$r_t = F_t * r_{t-1} + I_t * c_t \tag{4}$$

At last, the output gate O_t (Eq. 5) values is decided with the help of sigmoid layer. To do so, the c_t value is passed with *tanh* function and multiplied with Sigmoid activation function.

$$O_t = \sigma(W_O[H_{t-1}, i_t] + B_O) \tag{5}$$

$$x_t = O_t * tanh(c_t) \tag{6}$$

This research uses a LSTM model with and without adding the dropout (regularization parameter) to test the prediction accuracy. The experimental results obtained using these model settings are discusses in the result section V

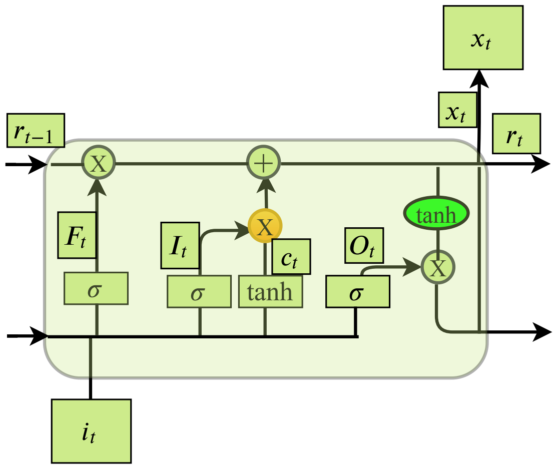


FIGURE 2: A unit of LSTM model [42]

C. MODEL SETTINGS

This section highlights the experimental setup and model evaluation technique that we followed to detect *HS* tweets for the dataset mentioned in section IV.

1) Experimental Setup

All experiments of this research carried out with Keras [44] with Tensorflow [45] backend. We have also used the *NumPy* [46], *NLTK* [47], *Scikit-learn* [48] to develop the models. To train the deep neural model, the dataset was preprocessed as it contained a large number of unwanted characters. The preprocessed dataset is split into two parts into a 3:1 ratio where three parts of the dataset are used to train the model whereas the fourth part is used to test. The developed DCNN model free from manual feature engineering and hence; we haven't extracted any manual feature from the dataset for this work.

The CNN and LSTM model run for 50 epochs for training with a batch size of 50 tweets. We have experimented with the CNN model by varying the size of tweets as 30, 40, and 50 and found it is not affected the model performance. Hence the size of each tweet fixed to 30 words. The tweets were having lesser than 30 words, padded with Zero to make it equal. Similarly, tweets having more than 30 words, the extra words are truncated. As shown in Figure 1, the first layer is the embedding layer. GloVe (a pre-trained word vector) is used to initialize the weight of the embedding layer [41]. Most of the parameters of the CNN model are set to default, few parameter's values set after the tuning. With the different value of dropout such as 0.2, 0.25, 0.3, 0.4, 0.5 model was experimented and found that 0.5 is yielded the best performance. Hence, the Dropout value is set to 0.5. This research treated *HS* problem as binary classification and hence uses softmax activation function at the output layer to predict the probability of input tweets with respect to *HS* and *NHS*. After tuning the model, the Rmsprop optimizer yielded better performance compared to Adam and Adaboost optimizer. Binary cross-entropy is used as a loss function.

D. EVALUATION TECHNIQUE

To evaluate the performance of the model for *HS* detection, researchers used the standard classification metrics, namely: Precision, Recall, and F1-score. Precision is defined as the number of *HS* tweets predicted truly among all *HS* tweets. Recall is defined as truly predicted *HS* tweets among all ground truth *HS* tweets. The F1-score is the harmonic mean of the Precision and Recall. These metrics are used to measure both the traditional machine learning based classifiers as well as the deep learning based models. The researchers also used the micro and macro average of precision, recall, and F1-score to represent the model performance. The micro average calculated by adding the true positive, false negative, and false-positive instances of *HS* predicted by the model, which is independent of the number of classes. The macro average is calculated by taking the average of calculated precision, recall, and F1-score for different classes. In previous studies, the researchers presented the model performance using the micro average precision, recall, and F1-score [9], [11], [12], [14], [31]. As mentioned, the statistics of our dataset in section IV, the dataset is imbalanced. The number of *HS* related tweets are very less compared to the *NHS* tweets. For such type of dataset, if one uses the micro-average based metrics, then it is very hard to find the performance of the minority class [13]. So, for an application like *HS* detection, the micro-average based metrics do not reflect the actual model performance. The above observation motivated us to use the standard version of Precision (Eq. 7), Recall (Eq. 8) and F1-score (Eq. 9) for evaluation of our proposed model, these metrics are defined as follows:

- Precision (*P*): It is the fraction of *HS* tweets among the retrieved *HS* tweets. It is computed as:

$$Precision = \frac{True_{Pos}}{True_{Pos} + False_{Pos}} \tag{7}$$

  Here $True_{Pos}$ means *HS* tweets predicted as *HS* and $False_{Pos}$ means the *NHS* tweets predicted as *HS*.
- Recall (*R*): It is the fraction of *HS* tweets that have been identified from the total number of *HS* tweets present.

$$Recall = \frac{True_{Pos}}{True_{Pos} + False_{Neg}} \tag{8}$$

  Here $False_{Neg}$ means *HS* tweets predicted as *NHS* by the model.
- *F1*-Score (*F1*): It is the harmonic mean of *P* (Eq. 7) and *R* (Eq. 8).

$$F1 - Score = 2 * \frac{P \, * \, R}{P \, + \, R} \tag{9}$$

V. RESULTS

The experimental results obtained using the different models will be discussed in this section. First, the results obtained using the baseline models will be discussed, and then the results of deep learning models will be discussed. For baseline model, seven different classifiers namely: (i) Logistic Regression (LR) [49], (ii) Naive Bayes (NB) [50], (iii) Random Forest (RF) [51], (iv) Support Vector Machine (SVM), (v) Decision Tree (DT), (vi) Gradient Boosting (GB) and (vii) K-nearest neighbour (KNN) are used. For the deep learning model, the DCNN, LSTM and C-LSTM model are used.

A. BASELINE MODELS

The experiment was started with traditional machine learning based classifiers. The required features were extracted from the tweet text using *tf-idf* technique by varying the *n*-gram. Five different set of features are extracted such as (i) uni-gram (1,1), (ii) bi-gram (2,2), (iii) bi-gram, tri-gram (2,3), (iv) bi-gram, quad-gram (2,4), and (v) bi-gram, five-gram (2,5). These features were fed into different classifiers.

First, the uni-gram (1,1) features fed into the selected classifiers. The best performance obtained using the RF classifier among all, where the prediction accuracy of *NHS* tweets is impressive, whereas for *HS* categories tweets, the prediction accuracy is very low. As shown in Table 2, the *P*, *R* and *F1* for *NHS* and *HS* categories tweets are 0.95, 1.00, 0.97 and 0.91, 0.17, 0.29 respectively for the best case. Next, we applied the same set of classifiers on other sets of features such as (2,2), (2,3), (2,4) and (2,5) grams. The experimental results with these sets of features are presented in Tables 3, 4, 5, and 6, the best results presented by bold text in every table. Among all selected classifiers, none of them are consistent. For every set of features, a new classifier performed better; For (1,1): RF, (2,2): RF, (2,3): SVM, (2,4): GB, and (2,5): KNN classifier yielded the best result. As can be seen from Figure 6,for all cases, the F1-score of *NHS* tweets is very high ($\approx$ 0.97), where as for *HS* related the best F1-score is 0.54. This results clearly indicated that the approximately half of the *HS* related tweets misclassified to *NHS* by the model. As shown in Figure 3, out of 581 *HS* test tweets, the model only predicted 235 tweets correctly, whereas 346 tweets are predicted as *NHS*. However, only 60 *NHS* tweets are misclassified out of 7410. The statistics shown in Figure 3 confirmed that the baseline model misclassified many *HS* related tweets.

There may be a list of reasons behind the low prediction rate for *HS* related tweets, first, the imbalanced nature of the data, the dataset used for this research contained 92.98% of *NHS* tweets and only 7.02% of *HS* tweets. Second, the lack of a semantic feature. The feature extracted using the *tf-idf* technique could not retain the contextual information of the tweets. This motivates us to use a model based on deep learning. The experimental results obtained from the deep learning models are discussed in section V-B.

B. RESULTS USING DEEP LEARNING MODEL

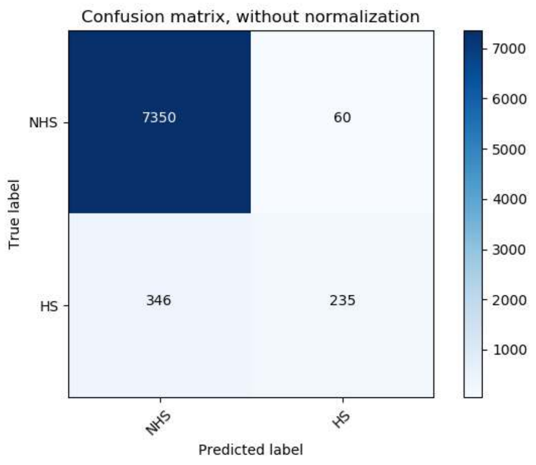
The traditional machine learning based SVM classifier achieved the best performance but limited to 0.80, 0.40, 0.54 values of precision, recall, and F1-score to detect *HS* tweets, which is not satisfactory. The majority of *HS* tweets are misclassified by the classifier. The deep learning based basic model of Convolutional Neural Network (1CNN) having a single layer of convolution and pooling layer yielded better performance (Table 7). The CNN model extracted the features by convolving the filters over the tweets' matrix. We varied the size of the filter from bi-gram to five-gram and captured the model performance. As shown in Table 7, the 1CNN model, varying the filter size, does not affect the model performance. In each case: i.e., 2g to 5g, the F1-score of the model for *HS* tweets is in the range of 0.57 to 0.59. However, this is 3% to 5% higher than that of the best

TABLE 2: Result using machine learning classifiers with n-gram (1, 1)g features

Classifier	Class	*P*	*R*	*F1*
LR	*NHS*	0.93	1.00	0.97
	HS	0.12	0.12	0.12
RF	*NHS*	0.94	1.00	0.97
	HS	**0.91**	**0.17**	**0.29**
NB	*NHS*	0.93	1.00	0.97
	HS	0.16	0.12	0.14
SVM	*NHS*	0.94	1.00	0.97
	HS	0.34	0.32	0.33
DT	*NHS*	0.95	0.94	0.94
	HS	0.24	0.28	0.26
GB	*NHS*	0.94	1.00	0.97
	HS	0.98	0.10	0.18
KNN	*NHS*	0.94	0.99	0.97
	HS	0.58	0.18	0.28

TABLE 3: Result using machine learning classifiers with n-gram (2g, 2g) features

Classifier	Class	*P*	*R*	*F1*
LR	*NHS*	0.94	1.00	0.97
	HS	0.78	0.19	0.31
RF	*NHS*	0.95	1.00	0.97
	HS	**0.85**	**0.30**	**0.44**
NB	*NHS*	0.93	1.00	0.96
	HS	1.00	0.04	0.08
SVM	*NHS*	0.94	1.00	0.97
	HS	0.80	0.20	0.32
DT	*NHS*	0.95	0.96	0.95
	HS	0.40	0.39	0.39
GB	*NHS*	0.94	1.00	0.97
	HS	0.84	0.22	0.35
KNN	*NHS*	0.95	0.99	0.97
	HS	0.75	0.28	0.41



s

FIGURE 3: Confusion Matrix obtained using SVM classifier

TABLE 4: Result using machine learning classifiers with n-gram (2g, 3g) features

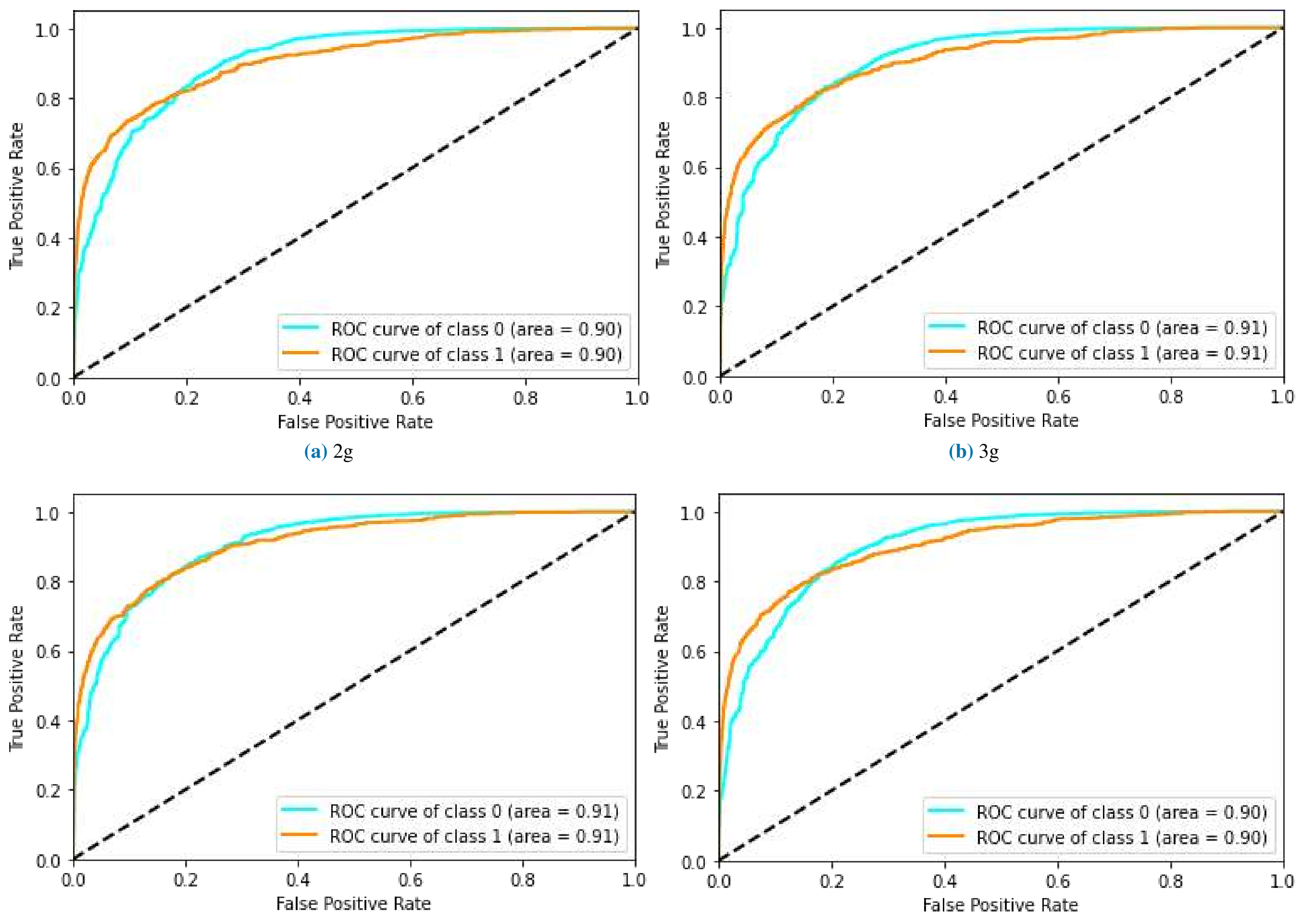
Classifier	Class	*P*	*R*	*F1*
LR	*NHS*	0.95	1.00	0.97
	HS	0.82	0.28	0.41
RF	*NHS*	0.95	0.99	0.97
	HS	0.66	0.38	0.48
NB	*NHS*	0.95	0.99	0.97
	HS	0.65	0.62	0.63
SVM	*NHS*	0.96	0.99	0.97
	HS	**0.80**	**0.40**	**0.54**
DT	*NHS*	0.96	0.99	0.97
	HS	0.70	0.40	0.51
GB	*NHS*	0.96	0.99	0.97
	HS	0.67	0.39	0.50
KNN	*NHS*	0.95	0.99	0.97
	HS	0.67	0.39	0.49

TABLE 5: Result using machine learning classifiers with n-gram (2g, 4g) features

Classifier	Class	*P*	*R*	*F1*
LR	*NHS*	0.95	1.00	0.97
	HS	0.84	0.25	0.39
RF	*NHS*	0.95	0.99	0.97
	HS	0.69	0.36	0.48
NB	*NHS*	0.96	0.98	0.97
	HS	0.64	0.38	0.48
SVM	*NHS*	0.95	0.99	0.97
	HS	0.73	0.39	0.51
DT	*NHS*	0.95	0.99	0.97
	HS	0.71	0.36	0.48
GB	*NHS*	0.96	0.99	0.97
	HS	**0.71**	**0.41**	**0.52**
KNN	*NHS*	0.96	0.99	0.97
	HS	0.70	0.40	0.51

TABLE 6: Result using machine learning classifiers with n-gram (2g, 5g) features

Classifier	Class	*P*	*R*	*F1*
LR	*NHS*	0.95	0.99	0.97
	HS	0.80	0.26	0.39
RF	*NHS*	0.96	0.99	0.97
	HS	0.56	0.40	0.46
NB	*NHS*	0.95	0.98	0.96
	HS	0.56	0.40	0.46
SVM	*NHS*	0.95	0.99	0.97
	HS	0.72	0.38	0.49
DT	*NHS*	0.95	0.99	0.97
	HS	0.74	0.36	0.49
GB	*NHS*	0.95	0.99	0.97
	HS	0.83	0.27	0.41
KNN	*NHS*	0.96	0.99	0.97
	HS	**0.73**	**0.39**	**0.51**



(a) 2g (b) 3g

(c) 4g (d) 5g

FIGURE 4: AUC-ROC curve using 1CNN with 2g-,3g-,4g- and 5g filter

TABLE 7: Result of 1CNN model with (2-5)g filters on GloVe embedding

Filter Size	Class	*P*	*R*	*F1*
2g	*NHS*	0.96	0.98	0.97
	HS	0.70	0.48	0.57
3g	*NHS*	0.96	0.99	0.97
	HS	0.72	0.49	0.58
4g	*NHS*	0.96	0.99	0.97
	HS	0.72	0.49	0.58
5g	*NHS*	0.97	0.97	0.97
	HS	**0.67**	**0.53**	**0.59**

results obtained using the SVM classifier (Table4). Hence, we can say, the CNN model performed better compared to the traditional ML-based models.

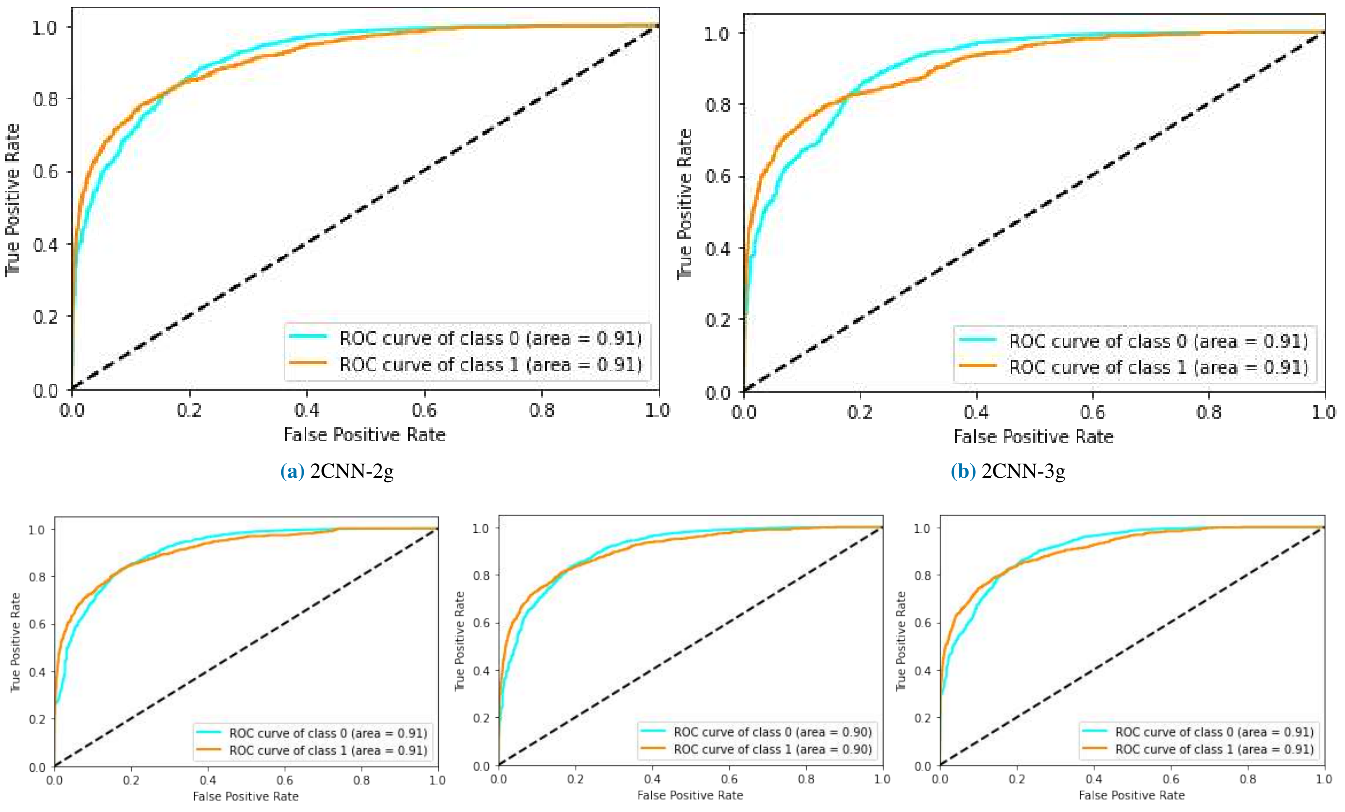
The best performance of the 1CNN model in terms of precision, recall, and F1-score is 0.65, 0.53, 0.59, respectively. In this case, also, the *HS* related tweets are not identified properly as the recall value limited to 0.53, which means only 53 *HS* tweets out of 100 detected successfully by the

TABLE 8: Result of 2CNN model with (2-5)g filters on GloVe embedding

Filter Size	Class	*P*	*R*	*F1*
2g	*NHS*	0.96	0.98	0.97
	HS	0.71	0.49	0.58
3g	*NHS*	0.96	0.99	0.97
	HS	0.71	0.48	0.57
4g	*NHS*	0.97	0.98	0.97
	HS	**0.65**	**0.53**	**0.59**
5g	*NHS*	0.97	0.97	0.97
	HS	0.60	0.54	0.56
(2-3-4)g	*NHS*	0.97	0.97	0.97
	HS	0.62	0.57	0.59

TABLE 9: Results of LSTM and C-LSTM model on GloVe embedding

Filter Size	Class	*P*	*R*	*F1*
LSTM	*NHS*	0.97	0.97	0.97
	HS	0.64	0.53	0.58
C-LSTM	*NHS*	0.96	0.99	0.97
	HS	0.75	0.43	0.55



(a) 2CNN-2g **(b)** 2CNN-3g **(c)** 2CNN-4g **(d)** 2CNN-5g **(e)** 2CNN-2-,3-,4g

FIGURE 5: AUC-ROC curve using 2CNN with 2g-,3g-,4g-,5g and 2-,3-,4g filters

model. However, for *NHS* related tweets, the model working well, the recall value for *NHS* tweets detection is 0.98; that means, only 2 *NHS* tweets are misclassified out of 100 *NHS* tweets. The AUC-ROC curves for different variants of 1CNN models are presented in Figures 4a, 4b,4c, 4d respectively. The experimental results confirmed that the overall accuracy of 1CNN models lie between 92-95%.

The experiment is repeated by adding a convolution layer in 1CNN model (say 2CNN). As shown in Table 8, the 2CNN model also has a similar performance as of 1CNN. The best recall value obtained using the 2CNN model is 0.53 for *HS* tweets prediction, which is similar to 1CNN (Figure 7).

Further, instead of using the individual n-gram filter for convolution operation, we applied 2-,3-, and 4-g filter collectively to extract the features from the tweets. The results obtained by applying 2-,3-, and 4-g filters collectively are presented in Table 10. In this case, the recall value of *NHS* tweets is 0.97 whereas for *HS* tweets it is 0.57 (greater than recall value obtained using individual filters). This model also failed to detect the 43 *HS* speech tweets out of 100. The AUC-ROC curves for different variants of 2CNN models are presented in Figures 5a, 5b,5c, 5d, 5e respectively. The accuracies of 2CNN models are also similar to 1CNN models, and it lies between 92-95%.

Another deep learning based model called Long Short-Term Memory (LSTM) is used to improve model performance. The working of LSTM model is explained in section IV-B. With the default setting of the hyper-parameters, LSTM model with 100 units of LSTM node yielded the 0.53 recall values, which is similar to 1CNN and 2CNN model performance. Further, the combination of CNN and LSTM: C-LSTM model is used. C-LSTM model having a similar configuration of 1CNN and an LSTM model (Table 9). The C-LSTM model provided the recall value of 0.43, which is lower than that of all tested models so far.

The LSTM model is generally performing well with the sequential data, where the model needs to preserve the semantics of the words for a long time; however, the *HS* detection issue is of the subjective type where the model has to process the complete tweet's words and extract their context. This may be one of the reasons behind the good performance of the CNN model that worked with complete words of the tweets to find their context.

By observing the results obtained using the traditional machine learning based classifiers and deep learning based models, we can say, the model with a fixed partition of train and test sample failed to achieve the satisfactory performance to detect the *HS* tweets. Finally, the *k*-fold cross-validation technique is used by setting the *k* value as 10. The 2CNN model having the filter size of 4g yielded the value of precision, recall and F1-score of 0.99, 0.99, 0.99 for *NHS* tweets prediction, and 0.97, 0.88, 0.92 for *HS* tweets predictions

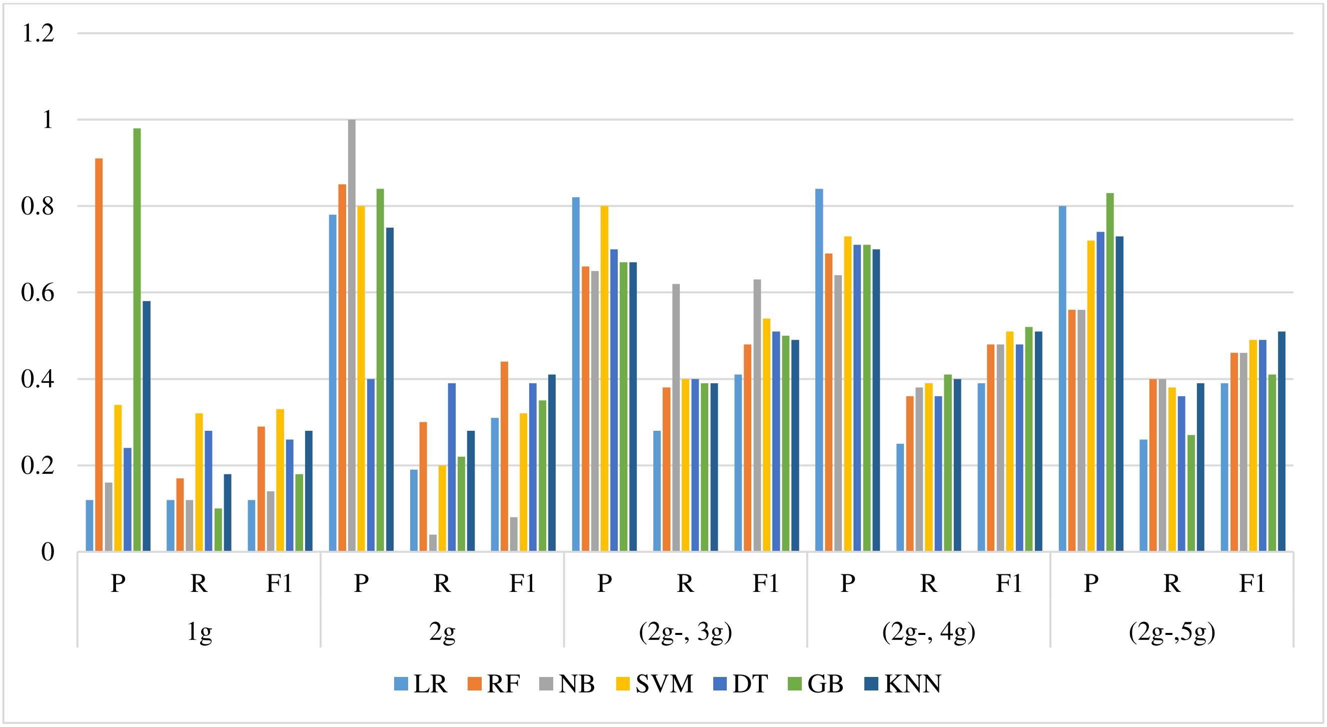


FIGURE 6: Performances of ML algorithms with different n-grams for *HS* predictions

TABLE 10: Results comparison of the proposed model with existing models

	Model	*P*	*R*	*F1*
Existing	Davidson et al. [8]	0.91	0.90	0.90
	Kamble and Joshi [32]	0.83	0.79	0.81
	Waseem and Hovy [23]	0.73	0.78	0.74
	Warner and Hirschberg [26]	0.68	0.60	0.63
Baseline	LR (2,3)	0.82	0.28	0.41
	RF (2,5)	0.73	0.41	0.53
	NB (2,4)	0.64	0.38	0.48
	SVM (2,3)	0.80	0.40	0.54
	DT (2,3)	0.70	0.40	0.51
	GB (2,4)	0.71	0.41	0.52
	KNN (2,4)	0.70	0.40	0.51
Proposed	1CNN (5g)	0.67	0.53	0.59
	2CNN (4g)	0.65	0.53	0.59
	2CNN (2-,3-,4)g	0.62	0.57	0.59
	C-LSTM	0.75	0.43	0.55
	LSTM	0.64	0.53	0.58
	DCNN with *k*-fold	**0.97**	**0.88**	**0.92**

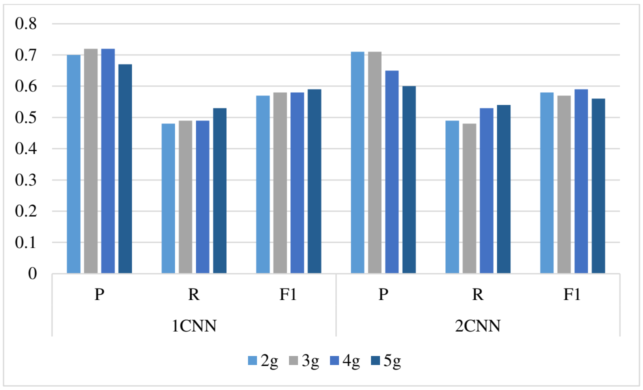


FIGURE 7: Performance of 1CNN and 2CNN model with varying size of kernels for *HS* predictions

(Table 10). It means, only one *NHS* tweets and twelve *HS* related tweets are misclassified out of 100. We compared the prediction accuracy with the existing model [8], [23], [26], [32] and found that the proposed model achieved the better prediction values as shown in Table 10.

VI. CONCLUSION, LIMITATION AND FUTURE SCOPE

This research addresses the issue of hate speech detection on Twitter using a deep convolutional neural network. Initially, the machine learning based classifiers such as LR, RF, NB, SVM, DT, GB, and KNN were used to identify the *HS* related tweets on Twitter with the features extracted using *tf-idf* technique. However, the best ML model, i.e., SVM able to predict only 53% of *HS* tweets correctly on a 3:1 train-test dataset. The reason behind the low prediction of *HS* tweets may include the imbalanced dataset, hence the model biased towards the *NHS* tweets prediction as it is having the majority of instances. Deep learning based CNN, LSTM, and their combinations C-LSTM models also have similar results with the fixed partitioned dataset. The experimental outcome on both the traditional machine learning based models and deep learning based models confirmed that none of the models predicted the *HS* tweets with satisfactory accuracy on a fixed partitioned of train-test. Finally, 10-fold cross-validation was used with the proposed DCNN model and achieved the best prediction recall value of 0.88 for *HS* and 0.99 for *NHS*. The experimental results confirmed the *k*-fold cross-validation technique is a better choice with the imbalanced dataset. The current research addressed the *HS* issues with the textual data only; however, images also widely used for the same. Hence, in the future, the researcher may include images with text or can analyse the video dataset to capture more *HS* related posts from Twitter. This research only used the tweets written in the English language, which can be further extended by mixing other languages such as Japanese, Hindi, Tamil, etc. The developed model achieved the recall value of 0.88, which indicates few tweets are not detected properly. In the future, a model may develop, which captures all hate speech content from the OSN. To build a general framework using deep learning models, the training dataset must have sufficient samples, in the future, the current dataset may be extended to achieve better accuracy.

REFERENCES

[1] T. K. Das, D. Acharjya, and M. Patra, “Opinion mining about a product by analyzing public tweets in twitter,” in 2014 International Conference on Computer Communication and Informatics. IEEE, 2014, pp. 1–4.
[2] H. Watanabe, M. Bouazizi, and T. Ohtsuki, “Hate speech on twitter: A pragmatic approach to collect hateful and offensive expressions and perform hate speech detection,” IEEE Access, vol. 6, pp. 13 825–13 835, 2018.
[3] O. Oriola and E. Kotzé, “Evaluating machine learning techniques for detecting offensive and hate speech in south african tweets,” IEEE Access, vol. 8, pp. 21 496–21 509, 2020.
[4] K. Weller, A. Bruns, J. Burgess, M. Mahrt, and C. Puschmann, Twitter and society. P. Lang, 2014, vol. 89.
[5] F. Del Vigna12, A. Cimino23, F. Dell’Orletta, M. Petrocchi, and M. Tesconi, “Hate me, hate me not: Hate speech detection on facebook,” in Proceedings of the First Italian Conference on Cybersecurity (ITASEC17), 2017, pp. 86–95.
[6] C. Stokel-Walker, “Alt-right’s’ twitter’is hate-speech hub,” New scientist, no. 3167, p. 15, 2018.
[7] P. Charitidis, S. Doropoulos, S. Vologiannidis, I. Papastergiou, and S. Karakeva, “Towards countering hate speech against journalists on social media,” Online Social Networks and Media, vol. 17, pp. 1–10, 2020.
[8] T. Davidson, D. Warmsley, M. Macy, and I. Weber, “Automated hate speech detection and the problem of offensive language,” in Eleventh international aaai conference on web and social media, 2017, pp. 512–515.
[9] Z. Waseem, “Are you a racist or am i seeing things? annotator influence on hate speech detection on twitter,” in Proceedings of the first workshop on NLP and computational social science, 2016, pp. 138–142.
[10] L. Gao and R. Huang, “Detecting online hate speech using context aware models,” arXiv preprint arXiv:1710.07395, pp. 260–266, 2017.
[11] P. Badjatiya, S. Gupta, M. Gupta, and V. Varma, “Deep learning for hate speech detection in tweets,” in Proceedings of the 26th International Conference on World Wide Web Companion, 2017, pp. 759–760.
[12] B. Gambäck and U. K. Sikdar, “Using convolutional neural networks to classify hate-speech,” in Proceedings of the first workshop on abusive language online, 2017, pp. 85–90.
[13] Z. Zhang and L. Luo, “Hate speech detection: A solved problem? the challenging case of long tail on twitter,” Semantic Web, vol. 10, no. 5, pp. 925–945, 2019.
[14] Z. Zhang, D. Robinson, and J. Tepper, “Detecting hate speech on twitter using a convolution-gru based deep neural network,” in European semantic web conference. Springer, 2018, pp. 745–760.
[15] Y. Kim, “Convolutional neural networks for sentence classification,” arXiv preprint arXiv:1408.5882, 2014.
[16] H. Sadr, M. M. Pedram, and M. Teshnehlab, “A robust sentiment analysis method based on sequential combination of convolutional and recursive neural networks,” Neural Processing Letters, pp. 1–17, 2019.
[17] P. K. Roy, “Multilayer convolutional neural network to filter low quality content from quora.” Neural Process Letters, pp. 1–17, 2020.
[18] M. P. Akhter, Z. Jiangbin, I. R. Naqvi, M. Abdelmajeed, A. Mehmood, and M. T. Sadiq, “Document-level text classification using single-layer multisize filters convolutional neural network,” IEEE Access, vol. 8, pp. 42 689–42 707, 2020.
[19] J. P. Dordevic, “The sociocognitive dimension of hate speech in readers’ comments on serbian news websites,” Discourse, Context & Media, vol. 33, p. 100366, 2020.
[20] J. Zheng and L. Zheng, “A hybrid bidirectional recurrent convolutional neural network attention-based model for text classification,” IEEE Access, vol. 7, pp. 106 673–106 685, 2019.
[21] H. M. Saleem, K. P. Dillon, S. Benesch, and D. Ruths, “A web of hate: Tackling hateful speech in online social spaces,” arXiv preprint arXiv:1709.10159, 2017.
[22] E. Wulczyn, N. Thain, and L. Dixon, “Ex machina: Personal attacks seen at scale,” in Proceedings of the 26th International Conference on World Wide Web, 2017, pp. 1391–1399.
[23] Z. Waseem and D. Hovy, “Hateful symbols or hateful people? predictive features for hate speech detection on twitter,” in Proceedings of the NAACL student research workshop, 2016, pp. 88–93.

[24] S. Sharma, S. Agrawal, and M. Shrivastava, "Degree based classification of harmful speech using twitter data," arXiv preprint arXiv:1806.04197, 2018.

[25] I. Kwok and Y. Wang, "Locate the hate: Detecting tweets against blacks," in Twenty-seventh AAAI conference on artificial intelligence, 2013, pp. 1621–1622.

[26] W. Warner and J. Hirschberg, "Detecting hate speech on the world wide web," in Proceedings of the Second Workshop on Language in Social Media. Montréal, Canada: Association for Computational Linguistics, Jun. 2012, pp. 19–26.

[27] P. Burnap and M. L. Williams, "Cyber hate speech on twitter: An application of machine classification and statistical modeling for policy and decision making," Policy & Internet, vol. 7, no. 2, pp. 223–242, 2015.

[28] N. Djuric, J. Zhou, R. Morris, M. Grbovic, V. Radosavljevic, and N. Bhamidipati, "Hate speech detection with comment embeddings," in Proceedings of the 24th international conference on world wide web, 2015, pp. 29–30.

[29] T. Joachims, "Making large-scale svm learning practical," Technical Report, Tech. Rep., 1998.

[30] Z. Waseem and D. Hovy, "Hateful symbols or hateful people? predictive features for hate speech detection on twitter," in Proceedings of the NAACL Student Research Workshop. San Diego, California: Association for Computational Linguistics, Jun. 2016, pp. 88–93.

[31] J. H. Park and P. Fung, "One-step and two-step classification for abusive language detection on twitter," arXiv preprint arXiv:1706.01206, pp. 41–45, 2017.

[32] S. Kamble and A. Joshi, "Hate speech detection from code-mixed hindi-english tweets using deep learning models," arXiv preprint arXiv:1811.05145, 2018.

[33] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," arXiv preprint arXiv:1404.2188, 2014.

[34] T. He, W. Huang, Y. Qiao, and J. Yao, "Text-attentional convolutional neural network for scene text detection," IEEE Transactions on Image Processing, vol. 25, no. 6, pp. 2529–2541, 2016.

[35] J. Wang, Y. Li, J. Shan, J. Bao, C. Zong, and L. Zhao, "Large-scale text classification using scope-based convolutional neural network: A deep learning approach," IEEE Access, vol. 7, pp. 171 548–171 558, 2019.

[36] M. Alali, N. Mohd Sharef, M. A. Azmi Murad, H. Hamdan, and N. A. Husin, "Narrow convolutional neural network for arabic dialects polarity classification," IEEE Access, vol. 7, pp. 96 272–96 283, 2019.

[37] C. Liu, W. Hsaio, and Y. Tu, "Time series classification with multivariate convolutional neural network," IEEE Transactions on Industrial Electronics, vol. 66, no. 6, pp. 4788–4797, 2019.

[38] X. Ouyang, K. Gu, and P. Zhou, "Spatial pyramid pooling mechanism in 3d convolutional network for sentence-level classification," IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 26, no. 11, pp. 2167–2179, 2018.

[39] P. K. Roy and J. P. Singh, "Predicting closed questions on community question answering sites using convolutional neural network," Neural Computing ansd Applications, pp. 1–18, 2019.

[40] A. Kumar and J. P. Singh, "Location reference identification from tweets during emergencies: A deep learning approach," International journal of disaster risk reduction, vol. 33, pp. 365–375, 2019.

[41] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014, pp. 1532–1543.

[42] P. K. Roy, J. P. Singh, and S. Banerjee, "Deep learning to filter sms spam," Future Generation Computer Systems, vol. 102, pp. 524–533, 2020.

[43] A. Kumar, J. P. Singh, Y. K. Dwivedi, and N. P. Rana, "A deep multi-modal neural network for informative twitter content classification during emergencies," Annals of Operations Research, pp. 1–32, 2020.

[44] F. Chollet et al., "Keras," https://keras.io, 2015.

[45] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[46] T. E. Oliphant, A guide to NumPy. Trelgol Publishing USA, 2006, vol. 1.

[47] S. Bird, E. Klein, and E. Loper, Natural Language Processing with Python. O'Reilly Media, 2009.

[48] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg et al., "Scikit-learn: Machine learning in python," Journal of machine learning research, vol. 12, no. Oct, pp. 2825–2830, 2011.

[49] N. M. Nasrabadi, "Pattern recognition and machine learning," Journal of electronic imaging, vol. 16, no. 4, p. 049901, 2007.

[50] I. Rish, "An empirical study of the naive bayes classifier," in IJCAI 2001 workshop on empirical methods in artificial intelligence, vol. 3, no. 22. IBM, 2001, pp. 41–46.

[51] L. Breiman, "Random forests," Machine learning, vol. 45, no. 1, pp. 5–32, 2001.