The Journal of Supercomputing
https://doi.org/10.1007/s11227-021-03838-w

A novel LSTM–CNN–grid search-based deep neural network for sentiment analysis

Ishaani Priyadarshini[1] · **Chase Cotton**[1]

Accepted: 21 April 2021

Abstract
As the number of users getting acquainted with the Internet is escalating rapidly, there is more user-generated content on the web. Comprehending hidden opinions, sentiments, and emotions in emails, tweets, reviews, and comments is a challenge and equally crucial for social media monitoring, brand monitoring, customer services, and market research. Sentiment analysis determines the emotional tone behind a series of words may essentially be used to understand the attitude, opinions, and emotions of users. We propose a novel long short-term memory (LSTM)–convolutional neural networks (CNN)–grid search-based deep neural network model for sentiment analysis. The study considers baseline algorithms like convolutional neural networks, *K*-nearest neighbor, LSTM, neural networks, LSTM–CNN, and CNN–LSTM which have been evaluated using accuracy, precision, sensitivity, specificity, and F-1 score, on multiple datasets. Our results show that the proposed model based on hyperparameter optimization outperforms other baseline models with an overall accuracy greater than 96%.

Keywords Sentiment analysis · Deep neural network · Long short-term memory (LSTM) · Convolutional neural networks (CNN) · Grid search

1 Introduction

Sentiment analysis is one of the most common and challenging problems in artificial intelligence. It uses automated tools for detecting subjective information like opinions, attitudes, and feelings expressed in text [1, 2] and is applied across news, blogs, and social networks [3, 4]. Sentiment analysis detects polarity (positive or negative

✉ Ishaani Priyadarshini
ishaani@udel.edu

Chase Cotton
ccotton@udel.edu

[1] Department of Electrical and Computer Engineering, University of Delaware, Newark, USA

Published online: 05 May 2021

opinion) within the text, which may be in the form of a document, paragraph, sentence, or clause [5]. It finds use in businesses so that the sentiments of customers toward a product may be identified. Similarly, performing sentiment analysis on survey responses and social media conversations may assist brands in identifying issues and tailoring their products according to customers' needs [6]. Sentiment analysis can help identify critical issues in real time. The smart city research community recognizes sentiment analysis as one of the best approaches to understand reactions and needs of people and concerns of the city governments [7]. Smart cities are driven immensely by big data and real-time analytics social media mining and sentiment analysis may lead to amplifying voices of citizens in smart cities. Based on the current state of art, there is relatively little known about how to best harness the potential benefits for smart cities regarding sentiment analysis and opinion mining. Hence, there is a need to strengthen the concept with respect to previously available research works. Since the internet has a plethora of data and smart cities are driven by the internet [8, 9], there is a need to sort data in a cost-effective and efficient way which is taken care of by sentiment analysis. There are many types of sentiment analysis like fine-grained sentiment analysis (very positive, positive, neutral, negative, very negative), emotion detection, aspect-based sentiment analysis (positive, negative), and multilingual sentiment analysis. The internet is full of applications, services, websites, blogs, social media platforms etc., which have enormous quantities of subjective texts. Subjective texts incorporate statements that have emotions, feelings and moods, and are often used in analyzing sentiments pertaining to social media, businesses, movies, product launch, etc. Sentiment analysis is performed by identifying positive and negative sentences based on their polarities. A negative sentence incorporates negative words like not, never, no, nobody, etc., while a positive sentence does not include any negative words. Polarity is used to quantify these sentiments by assigning polarity signs and scores. Human language being so elaborate with nearly infinite grammatical variations, misspellings, slang, and other challenges, makes automated analysis of natural language difficult. Complicated sentence structures in the English language are also to be blamed for the same. There may exist ambiguity in keyword definitions, such that words can change their meanings with respect to various usages and contexts. It may also be difficult for a system to recognize sentences without keywords; thus, sentences without keywords would simply imply that there are no emotions. Sometimes, emotions in text messages rely on syntax structures and semantics; thus, ignoring linguistic information may also result in misclassification. Finally, determining emotion indicators is a tedious task. All these reasons call for sophisticated machine learning techniques that can be utilized for performing sentiment analysis. Owing to the limitations of sentiment analysis, it has been proposed as a means to distinguish machines and humans in the past [10–12]. Hence, sophisticated sentiment analysis approaches may bridge the gap between humans and machines, and make sentiment analysis a simpler problem. Past research works on sentiment analysis have relied on many machine learning algorithms [13–16]. In this study, we explore the sentiment analysis of sentences using a few more artificial intelligence techniques. We also propose a novel long short-term memory–convolutional neural network–grid search-based deep neural network for identifying sentences. The main objective of incorporating grid search

into the LSTM–CNN model is for hyperparameter optimization. Hyperparameters are values that can control the process of learning. Tuning hyperparameters ensure that the model can optimally solve a problem by minimizing the pre-defined losses and giving accurate results. For this study, we use some baseline artificial intelligence algorithms, i.e., CNN, *K*-NN, LSTM, neural networks, CNN–LSTM, and LSTM–CNN. The techniques have been evaluated using evaluation parameters like accuracy, precision, sensitivity, specificity, and F-1 score on multiple datasets.

The rest of the paper is organized as follows. Section 2 lists the materials and methods involved in the study. Here, we discuss the related research works and artificial intelligence techniques that we have used for the study, along with the proposed model. Section 3 includes the results based on experimental analysis and evaluation parameters. Section 4 considers the discussions based on the results obtained along with a comparative analysis. Section 5 concludes the study and highlights Future Works.

2 Materials and methods

In this section, we describe the related works and artificial intelligence techniques supporting the work. We also highlight the proposed method and the overall methodology of the proposed work.

2.1 Related works

Basiri et al. [17] suggested sentiment analysis based on an attention-based bidirectional CNN–RNN deep model. The study has been conducted using both past and future contexts. Five reviews and three twitter datasets were considered for the study. Jin et al. [18] suggested sentiment analysis based on heterogeneous graph network embedding based on variational auto-encoder for learning joint representations of users' social relationship. This is encouraged by preserving structural proximity and attribute proximity, respectively. The model conveniently outperforms traditional text-based sentiment analysis approaches [19] recommended a Bidirectional Encoder Representations from Transformers (BERT) based pipeline for Twitter sentiment analysis. The study is interesting as it aims to transform jargons into plaintext, and the tweets are classified using BERT but pre-trained on plaintext. The model is applicable to many languages. Lu et al. [20] performed an aspect-based sentiment analysis using aspect-gated graph convolutional networks. The model makes use of syntactical information and sentiment dependencies, and the experimental analysis has been conducted over multiple SemEval datasets. The proposed model outperformed baseline models with an increase of 2.14% and 1.33% in accuracy and Macro-F1, respectively. Nemes and Kiss [21] performed a sentiment analysis of social media based on COVID 19 (comments, hashtags, posts, tweets). While COVID-19 outbreak had an effect over the world [22, 23], the study considered recurrent neural networks (RNN) for the analysis. The study concluded that there are more positive tweets over social media. Tubishat et al. [24] relied on optimal rules

combination for explicit aspects extraction in sentiment analysis. The algorithm presented incorporates 126 aspect extraction rules for both formal and informal texts, which primarily consider dependency based rules and pattern based rules. The study also improvises whale optimization algorithm to address issues related to rules selection problem. Kandasamy et al. [25] presented yet another approach for sentiment analysis using neutrosophic sets, such that seven membership functions can be considered from the primarily existing three functions [26–28]. The study shows that multi-refined neutrosophic sets perform well in analyzing the sentiments from texts. Huang et al. [29] suggested sentiment analysis based on attention–emotion-enhanced convolutional LSTM to address the issue of high level abstractions. The LSTM network in the study is improvised by incorporating emotional intelligence and attention mechanisms. The model is also supported by convolution, pooling and concatenation, and manifests appreciable performance. Zhao et al. [30] presented a combination of CNN and gated recurrent networks (GRU) for sentiment analysis. The proposed model relies on local features generated by CNN and the long-term dependencies learned by GRU. Experimental analysis on multiple datasets validates the robustness of the proposed model. Srividya and Sowjanya [31] recommended aspect-based sentiment analysis by employing a neural attention-based model. The model has been trained on datasets Rest14, Rest15, and Rest16, and the performance has been evaluated based on accuracy and F-1 score.

Table 1 depicts various methodologies adopted in the past for sentiment analysis.

2.2 Artificial intelligence techniques

In this section, we present some artificial intelligence techniques that we considered for the study. These techniques are also the baseline algorithms with which we compare our proposed model.

Table 1 Sentiment analysis methodologies based on past research works

References	Methodology
Basiri et al. [17]	Attention-based bidirectional CNN–RNN deep model
Jin et al. [18]	Heterogeneous graph network embedding
Pota et al. [19]	BERT-based pipeline
Lu et al. [20]	Aspect-gated graph convolutional networks
Nemes and Kiss [21]	Recurrent neural networks
Tubishat et al. [24]	Explicit aspects extraction using optimal rules combination
Kandasamy et al. [25]	Refined neutrosophic sets
Huang et al. [29]	Attention–emotion-enhanced convolutional LSTM
Zhao et al. [30]	Combination of convolutional neural network and gated recurrent unit
Srividya and Sowjanya [31]	Neural attention-based model

2.2.1 Convolutional neural networks (CNN)

Convolutional neural networks (CNN), also referred to as covnets, are neural networks that may have common parameters. A covnets consists of a series of layers, such that each layer is capable of transforming one volume to another through differentiable function [32, 33]. There are various types of layers involved in CNN. The input layer holds the raw input of the image while the convolution layer is responsible for computing the output volume by performing the dot product operation between all filters and image patches. The activation function layer is responsible for applying element wise activation function to the output of the initial layer or the convolutional layer. Activation functions may be in the form of Sigmoid, ReLU, Leaky ReLU, tanh, Softmax, etc. Sigmoid is a nonlinear activation function which transforms the values between 0 and 1. If there are multiple neurons with sigmoid function as their activation function, the output obtained will also be nonlinear. The ReLU function or the rectified linear unit is yet another nonlinear activation function. Unlike other activation functions, ReLU does not activate all the neurons at the same time. Hence, the neurons get deactivated only if the output of the linear transformation is less than 0. Leaky ReLU function is t an improved version of the ReLU function. In ReLU, the gradient is 0 for $x<0$, which leads to deactivation of neurons in that region. This problem is addressed by Leaky ReLU by defining ReLU function as an extremely small linear component of x. The tanh function is very similar to the sigmoid function except that it is symmetric around the origin such that the values range from -1 to 1. Hence, the inputs to the next layers will not always be of the same sign. Softmax function may be thought of as multiple sigmoids; hence, it is widely used in binary classification problems. The Softmax function is also applicable to multiclass classification problems. Pool layer is accountable for reducing the size of volume and increasing the computational efficiency. It is inserted in the covnets and its main objective is preventing any kind of overfitting. Pooling layers can be either max pooling or average pooling. Fully connected layer is a regular neural network layer which takes input from the previous layer. Its main objective is to compute the class scores so that it can output the 1-D array of size equal to the number of classes.

2.2.2 *K*-nearest neighbor (*K*-NN)

K-nearest neighbors is one of the most fundamental yet basic classification techniques in machine learning. It belongs to the family of supervised learning and finds exceptional application in pattern recognition, knowledge discovery of data and intrusion detection. It is generally dispensable, in actuality, situations since it is nonparametric, which means, it does not make any fundamental presumptions about the appropriation of information. For sentiment analysis, *K*-NN algorithm performs classification by finding the *K*-nearest matches in training data. *K*-NN is based on the principle that assumes that every data point falling near to each other falls in the same class. Hence, it is capable of classifying a new data point based on similarity. *K*-NN relies on 'feature similarity' for predicting the values of new data points. This means that the new data point will be assigned a value based on how closely it

matches the points in the training set. This is followed by using the label of closest matches for prediction [34]. The main advantage of this technique is that it is quite easy to implement and is robust with respect to search space. *K*-NN is useful for handling nonlinear data since there is no assumption about data in this algorithm.

2.2.3 Long short-term memory (LSTM)

Long short-term memory is a sort of recurrent neural network. In RNN, yield from the last phase is taken care of as input to the current phase. LSTM was introduced by Hochreiter and Schmidhuber [35, 36]. It handled the issue of long-term dependencies of RNN in which the RNN cannot anticipate the word stored in the long-term memory, however, can give more precise predictions from the new data. As the gap length expands RNN is unable to provide effective performance. LSTM can retain the data for extensive stretches of time. It is utilized for preparing, classifying and predicting based on time series information. LSTM has a chain structure that contains four neural networks and distinctive memory blocks called cells. These cells are responsible for information retention, while gates are responsible for memory manipulations. The Forget gate ensures that information that is no longer useful is eradicated The input gate is responsible for adding useful information to the cell. The output gate is used for extracting useful information from cells.

2.2.4 Neural networks (NN)

In a normal neural network, there are three kinds of layers [37, 38]. The input to the model is given through this layer. The number of neurons in this layer is equivalent to the combined number of features in our data. The contribution from the input layer is then provided to the hidden layer. There can be many concealed layers relying on our model and data size. Every hidden layer may have various quantities of neurons which are commonly more prominent than the quantity of features. The yield from each layer is computed by matrix multiplication of output of the previous layer with learnable loads of that layer and afterward by addition of learnable biases followed by activation function. This is responsible for making the system nonlinear. The yield from the hidden layer is then fed into a logistic operation like sigmoid or Softmax which changes over the yield of each class into the likelihood score of each class. The information is then presented into the model and yield from each layer is acquired. This stage is called feedforward; we compute the error utilizing an error function, some of which are cross-entropy, square loss error, and so forth. From that point forward, we backpropagate into the model by ascertaining the derivatives. This progression is called backpropagation which essentially is utilized to limit the loss.

2.2.5 CNN–LSTM

A combination of CNN–LSTM architecture includes an initial convolution layer which is responsible for receiving word embeddings input. This process yields an output which is pooled to a smaller dimension and finally fed into an LSTM layer. The underlying idea behind the model is that the local features will be extracted by

the convolution later, such that the LSTM will order the features to comprehend the ordering of the input text [39]. Dropout is a method of ignoring randomly selected neurons during training. 'Dropping-out' randomly means that the contribution of the neurons to the activation of downstream neurons is temporarily withdrawn on the forward pass. Moreover, weight updates are no longer applied to the neuron on the backward pass. Dropout technique is used to prevent a model from overfitting. It works by randomly assigning the value zero to outgoing edges of hidden units during every update of the training phase. These hidden units are made up of neurons (Fig. 1).

2.2.6 LSTM–CNN

The LSTM–CNN model is more powerful as compared to the CNN–LSTM model. The LSTM–CNN architecture includes an initial LSTM layer which is responsible for receiving word embeddings for each token in the sentences as inputs. The underlying idea is that the output token will hold more information for the initial tokens as well as the previous tokens [35]. The LSTM layer in this model is accountable for generating new encoding for the original input [40]. The output generated from the LSTM layer is fed into the CNN, which is equipped with extracting local features. The output of this convolution layer is then pooled to a smaller dimension and the final output results as either a positive or a negative label (Fig. 2).

2.3 Proposed method

Our proposed work focuses on sentiment analysis of text using a novel LSTM–CNN–grid search-based deep neural network. Often, machine learning algorithms require choosing a set of optimal parameters, also known as tuning or hyperparameter optimization. Hyperparameters are parameters which can control

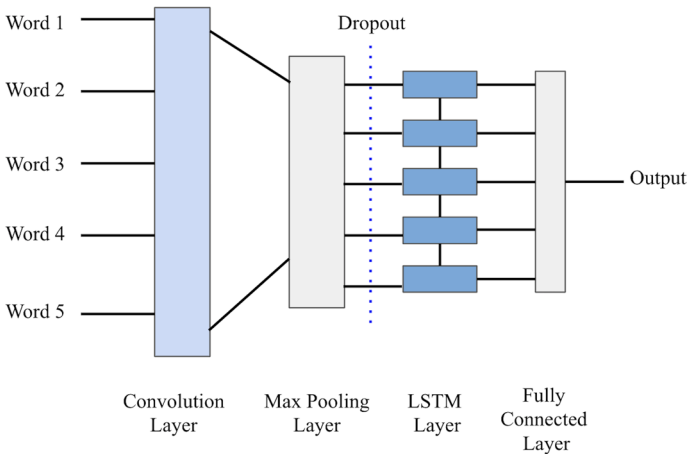

Fig. 1 CNN–LSTM architecture

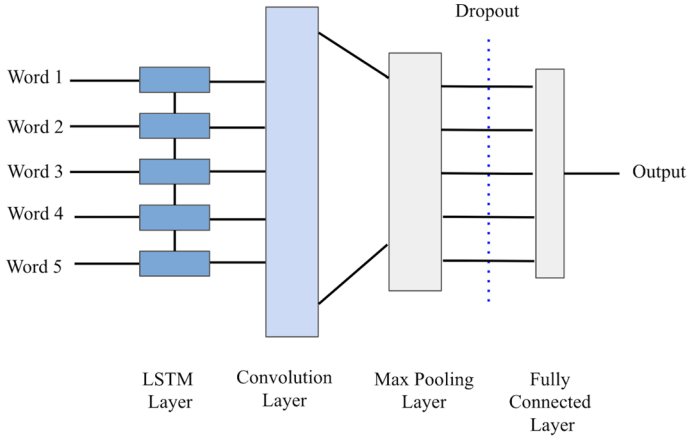

Fig. 2 LSTM–CNN architecture

the learning process. Similar machine learning algorithms may have different types of learning rates, weights, and constraints, for generalizing data patterns. These measures are called hyperparameters. It is necessary to tune hyperparameters so that the problem can be solved optimally. The process of optimization involves finding a tuple that provides an optimal model and minimizes loss function. There are several approaches to hyperparameter tuning. For our study, we have relied on the method of grid search, which consists of exhaustive searching through a subset of hyperparameter space of the algorithm, followed by a performance metric. In the grid searching process, data is scanned to configure optimal parameters of a specific model. Parameters are specific to the type of model considered. Grid searching is not confined to one type of model, but can be applied throughout machine learning, such that the best parameters can be identified for the model. The process of grid searching builds a model on each parameter combination possible, leading to multiple iterations. These model combinations for every parameter are stored, and hence, grid searching is computationally expensive. For a machine learner, parameter space may either include real valued or unbounded value spaces; hence, there may be a need to perform discretization [41].

In this proposed work, LSTM–CNN model is integrated with the fully connected neural network and grid search (Fig. 3). The main purpose behind the grid search is to locate optimal hyperparameters which are used to classify more accurate polarity of sentiments. The proposed architecture has several layers like the input, LSTM, convolution layer, max pooling layer, dropout, fully connected neural network, grid search, and output. The input passes through an LSTM layer with several units. As we know convolution refers to the mathematical combination of two functions which results in a third function, i.e., merging of two sets of information. In the case of a CNN, the convolution is performed on the input data using a filter or kernel, which further results in producing a feature map. The next layer is the max pooling layer. Maximum pooling refers to a pooling operation that determines the largest value in each patch of each feature map. Dropout is responsible for preventing the model

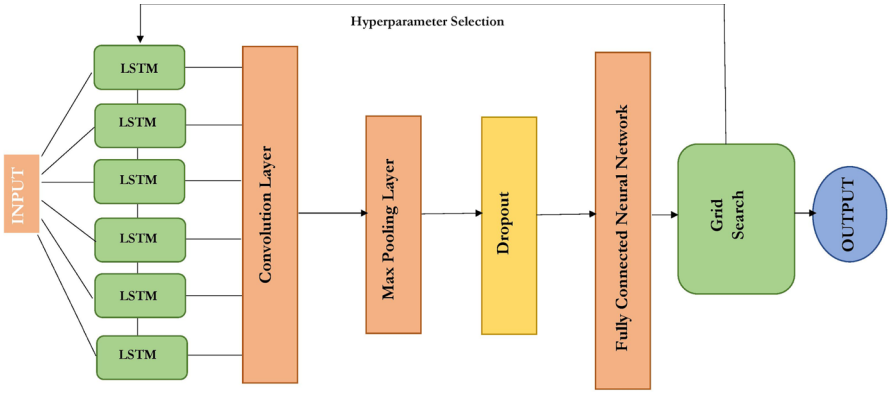

Fig. 3 Architecture of the proposed model

from overfitting. Fully connected layer is simply, feed forward neural networks forming the last few layers in the network. The output from the final pooling layer is flattened and fed into the fully connected layer; hence, it eventually becomes the input to the fully connected layer. Finally, we apply grid search to the overall operation. The output generated determines the sentiment of the texts. Hyperparameter tuning or optimization refers to choosing a set of optimal hyperparameters, which are set before training the machine. Two common methods of tuning the hyperparameters are grid search and random search. In grid search, every combination of a preset list of values of hyperparameters is tried, such that the best combination is chosen based on the cross-validation score. In random search, random combinations are chosen to train the model, such that the number of parameters to test can be controlled. Although it is efficient for testing a wide range of values and is capable of reaching a very good combination very fast, its major drawback is that it cannot guarantee the best parameters combination. Grid search, on the other hand, may take a lot of time, but will give the best combination. The hyperparameters employed in this model are illustrated in Table 2.

Table 2 depicts grid search hyperparameters considered for the study. For tuning hyperparameters, we have taken into account the batch size, epochs, optimization, drop regularization, and neurons in the hidden layer, respectively.

Table 2 Grid search hyperparameters

Hyperparameters	Values
Batch size	10, 20, 40, 60, 80, 100
Epcohes	10, 50, 100
Optimization	SGD, RMSprop, Adagrad, Adadelta, Adam, Adamax, Nadam
Drop regularization	0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9
Neurons in hidden layer	1, 5, 10, 15, 20, 25, 30

a. The batch size is a hyperparameter that gives an idea about the number of samples to be considered for tuning before the internal model parameters are updated.
b. The number of epochs is a hyperparameter that gives an estimate about the number times the learning algorithm will work for the training dataset. One epoch refers to each sample in the training dataset having a chance for updating the internal model parameters. An epoch may incorporate one or more batches. The number of epochs defines the time taken for the entire set of input training data to be passed through the neural network while training occurs. Multiple epochs during training will ensure that the network changes its weights during training. If too few or too many epochs are used, it may lead to underfitting or overfitting.
c. The optimization algorithm is yet another hyperparameter considered for tuning. It is possible for weights and biases of the network to change, while the algorithm trains, affecting the overall model performance. If network predictions based on the model are poor, the output of the loss function is high. An optimizer is responsible for bridging the gap between updating model parameters and the loss function. Here, loss function is used to indicate tuning quality. Stochastic Gradient Descent (SGD), Root Mean Square Propagation (RMSprop) Adagrad, Adadelta, Adam, Adamax, Nadam, etc., are some optimizers used in hyperparameter tuning.
d. Dropout method selectively chooses neurons to ignore during the training process. Once the contributions of these neurons are dropped, their weights are no longer applied. The underlying idea behind dropout is that, when a neural network trains, the weights of the neurons are adjusted with respect to the neurons. This can also lead to neighboring neurons becoming specialized, which may ultimately result in a model that suffers overfitting.
e. Neurons in a hidden layer may be defined as the number of units representing the number of neurons of a layer. A unit is responsible for taking inputs from all the nodes in the layer below. This is followed by calculation and the outcome is given as output to the layer above. The overall capacity of the network is controlled by the number of neurons in a layer.

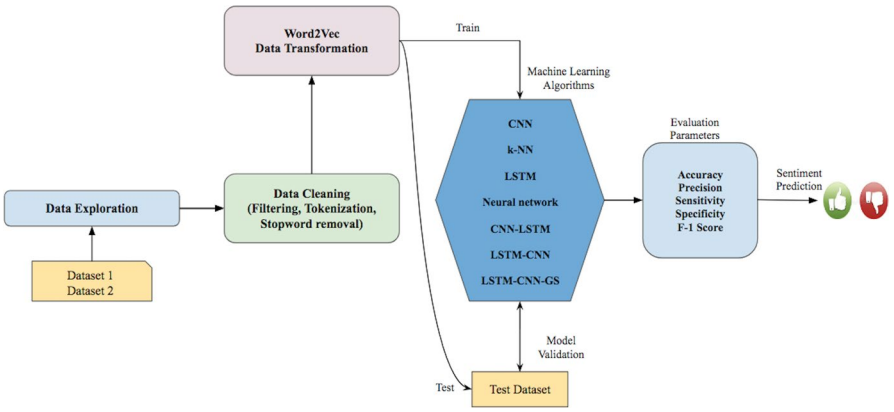

Fig. 4 Overall methodology of the proposed work

2.4 Methodology

The following figure (Fig. 4) represents the overall methodology of the proposed work. The process starts with data exploration, followed by data cleaning and data transformation, after which the data is split into training set and testing set. The classification models are validated using certain evaluation parameters.

The overall methodology of the proposed work incorporates a series of steps, which are detailed as follows:

1. *Data exploration* Data exploration refers to the initial data analysis in which statistical techniques and visualization methods are used for describing the characteristics of the datasets. This step provides us with basic information like size, quantity, accuracy etc., to understand the nature of the data. It can be performed manually or using software driven automated techniques. This step assists us in identifying relationships between different data variables, the structure of the dataset, the presence of outliers, and the distribution of data values. It can also provide information on patterns of data, which is initially gathered in large unstructured volumes.
2. *Data cleaning/preparation* Data preprocessing is a significant task in sentiment analysis. Preparing data simply means to bring the text into a form that is predictable and analyzable for the required task. In this step we convert raw messages into vectors. In order to achieve that we must follow certain steps. We often start by removing HTML (Hypertext Markup Language) and ASCII (American Standard Code for Information Interchange) characters. This is followed by converting text to lowercase, which is useful later while parsing. Removing punctuations is yet another important step for getting a vector representation of words. It is necessary to remove stopwords since they do not add much meaning to a sentence and can safely be ignored without sacrificing the meaning of the sentence. As we return a list of clean words, we also tokenize the message. Tokenizing a text refers to breaking characters into pieces, called as tokens, and simultaneously removing punctuations and stopwords. This is followed by stemming and lemmatization processes, which are responsible for generating the root form of the inflected words. While stemming considers the form of the word, lemmatization considers the meaning of the word.
3. *Data transformation* Following Tokenization, we perform Word Vectorization. In order to train our model on the corresponding data, there is a need to convert text data into numerical format. Word Vectorization is performed to map words or phrases from vocabulary to a corresponding vector of real numbers which used to find word predictions, word similarities/semantics. The process of converting words into numbers is called Vectorization. Word2vec is a method for efficient creation of word embeddings. It uses a two-layer neural network such that the input of word2vec is a text corpus and the output is a set of vectors (feature vectors) which reflect words in that corpus. Although Word2vec is not a deep neural network, it is capable of turning text into a numerical form that is easily comprehensible by deep neural networks. Word2Vec is responsible for causing the words with a similar context to have similar embeddings. It is based on continuous bag

of words (CBOW) and skip-gram model, and both the techniques involve learning weights that can act as word vector representations. CBOW and skip-gram model architectures are capable of learning the underlying word representations for each word by means of neural networks. The CBOW model witnesses distributed representations of context combined for predicting the word in the middle. On the other hand, for the skip-gram model, for predicting the context, the distributed representation of the input word is considered [42, 43].

4. *Applying machine learning algorithms* Once the process of data transformation is complete, the next step is to perform text classification using machine learning techniques. For this study, we have considered six baseline algorithms, i.e., CNN, *K*-NN, LSTM, neural networks, CNN–LSTM, and LSTM–CNN. Our proposed model is a variation of LSTM–CNN, such that there is an added component of hyperparameter tuning using grid search method, hence the name LSTM–CNN–GS.
5. *Evaluating the models* The data is split into training set and testing set, i.e., 80% training data and 20% test data. While the training set is used for building up the model, the test set is used to validate it. Hence, training data is considered for fitting the model, and test data is used for testing it. The machine learning algorithms classify the text into positive and negative sentences. The evaluation parameters considered for the study are accuracy, precision, sensitivity, specificity, and F-1 score, respectively.

3 Experimental analysis

In this section, we describe the datasets used for the study along with statistical parameters used for evaluating the models.

3.1 Datasets

The study has been performed on two datasets taken from Kaggle, i.e., Amazon reviews for sentiment analysis (Dataset 1) [44], and IMDB Dataset of 50K Movie Reviews (Dataset 2) [45]. Dataset 1 incorporates approximately four million Amazon customer reviews in the form of input text, while the star ratings are output labels. The customer reviews are actually significant from the business point of view. Moreover, the reviews have been separated into two classes for sentiment analysis, i.e., positive and negative reviews. X and Y are class names. Also, the classes are labeled Label1 and Label2, such that there is only one class per row. Label1 incorporates all 1-star and 2-star reviews, while Label2 denotes 4-star and 5-star reviews. The dataset does not incorporate 3-star reviews owing to neutrality. Also, the two classes are evenly balanced here. It is data written by users, so there is a possibility of various typos, non-standard spellings, and other variations. The positive sentences are denoted by 1, and the negative sentences are denoted by 0. The implementation has been performed using python. For training purposes, we have used 2,880,000 samples with 12,000 features, whereas for testing, we have used 720,000 samples with 12,000 features. Dataset 2 is

appropriate for binary sentiment classification and incorporates substantial amounts of data. The dataset has a maximum of 30 reviews per movie and has an even number of positive and negative reviews. The dataset includes highly polar movie reviews for training and for testing purposes. A negative review has a score less than equal to four out of ten, while a positive review has a score greater than equal to seven out of ten. The dataset does not include any neutral reviews. Hence, it is suitable for predicting the number of positive and negative reviews based on classification techniques and deep learning algorithms, which are a part of our analysis.

3.2 Statistical parameters

In the previous section, we mentioned the baselines algorithms as well as the proposed model for the study. In order to validate that our proposed model is efficient, we need to perform performance evaluation, which also happens to be a significant aspect of the machine learning process. For detecting whether a sentence is positive or negative, we have deployed several machine learning algorithms and artificial intelligence techniques. There is a need to determine model performance by means of evaluation. performance evaluation has been conducted using five parameters namely, accuracy, precision, sensitivity, specificity, and F1-Score, whose values may be determined using confusion matrix (Fig. 5).

a. *Accuracy* Accuracy is one of the most common metrics for evaluating classification models. It describes how frequently an algorithm classifies data correctly. Accuracy is described as the number of correctly predicted data points with respect to the total number of data points [46]. Given a confusion matrix, accuracy may be defined as the sum of True Negative and True Positives combined over the sum of True Negative (TN), True Positive (TP), False Negative (FN), and False Positive (FP).

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}. \tag{1}$$

b. *Precision* Precision is another popular metric that is used in classification, information retrieval, and pattern recognition-related problems. It is described as the number of relevant observations with respect to retrieved observations. Given a confusion matrix, precision may be calculated by True Positives with respect to the total number of True Positives and False Negatives combined.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{2}$$

Fig. 5 Confusion matrix

	Actual Positive	Actual Negative
Predicted Positive	True Positive (TP)	False Positive (FP)
Predicted Negative	False Negative (FN)	True Negative (TN)

c. *Sensitivity* Sensitivity is depicted by the ratio of actual positive events that got predicted as positive. It is also known as recall [37]. Given a confusion matrix, sensitivity may be calculated by True Positive value with respect to the sum of True Positive and False Negative combined)

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{3}$$

d. *Specificity* Specificity may be described as the ratio of actual negatives that got predicted as negative [47]. Given a confusion matrix, specificity would be calculated by True Negatives with respect to the sum of True Negatives and False Positives combined.

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}. \tag{4}$$

e. F1-Score: F-1 score is defined as the measure of a model's accuracy on a dataset and is popularly used to evaluate binary classification systems, which makes it apt for our study (positive and negative). It combines the values of precision and recall and is given by

$$\text{F-1 Score} = \frac{\text{TP}}{\text{TP} + 0.5(\text{FP} + \text{FN})}. \tag{5}$$

4 Results and discussion

Sentiment analysis is a challenging issue in Natural Language Processing. Using two diverse and unbiased datasets, we successfully implemented artificial intelligence techniques so as to identify positive and negative sentences. In this section, we present the results based on the experimental analysis.

4.1 Results

The performance evaluation for the artificial intelligence techniques has been conducted taking into account specific evaluation parameters like accuracy, precision, sensitivity, specificity, and F-1 score. Tables 3 and 4 depict the values of these parameters for the study conducted, which involved 80% training data and 20% test data.

For dataset 1 (Table 3), we observe that the *K*-NN values for the evaluation parameters are 0.779, 0.835, 0.867, 0.928, and 0.85, respectively. For neural networks, the performance values are 0.897, 0.90, 0.90, 0.915, and 0.90, respectively, whereas for CNN the values seem slightly better, i.e., 0.927, 0.922, 0.918, 0.913, and 0.928, respectively. LSTM performs reasonably well with values 0.905, 0.912, 0.897, 0.929, and 0.915. Integrated model CNN–LSTM demonstrates fair performance with values 0.945, 0.939, 0.925, 0.925, and 0.931, respectively. The performance

Table 3 Performance evaluation of Dataset 1

AI models	Accuracy	Precision	Sensitivity	Specificity	F-1 score
K-NN	0.779	0.835	0.867	0.928	0.85
NN	0.897	0.90	0.90	0.915	0.90
CNN	0.927	0.922	0.918	0.913	0.928
LSTM	0.905	0.912	0.897	0.929	0.915
CNN–LSTM	0.945	0.939	0.925	0.925	0.931
LSTM–CNN	0.958	0.943	0.933	0.949	0.939
LSTM–CNN–GS	**0.964**	**0.989**	**0.974**	**0.992**	**0.981**

Table 4 Performance evaluation of Dataset 2

AI models	Accuracy	Precision	Sensitivity	Specificity	F-1 score
K-NN	0.889	0.896	0.891	0.918	0.889
NN	0.892	0.918	0.911	0.92	0.911
CNN	0.927	0.918	0.92	0.927	0.92
LSTM	0.915	0.909	0.92	0.919	0.91
CNN–LSTM	0.938	0.929	0.936	0.92	0.92
LSTM–CNN	0.947	0.931	0.922	0.94	0.959
LSTM–CNN–GS	**0.978**	**0.982**	**0.989**	**0.99**	**0.972**

of CNN–LSTM may be attributed to the CNN–LSTM architecture which involves feature extraction on input data using CNN layers supported by LSTMs to encourage sequence prediction. Likewise, in LSTM–CNN architecture, output generated from the LSTM layer is fed into the CNN. LSTM–CNN exhibits a better performance than CNN–LSTM with values 0.958, 0.943, 0.933, 0.949, and 0.939. The proposed model or the LSTM–CNN–GS is an LSTM–CNN model with hyperparameter tuning. The hyperparameter tuning method used in the study is grid search, and we observe that the performance of LSTM–CNN–GS has outperformed all the other baseline models with values 0.964, 0.989, 0.974, 0.992, and 0.981, respectively. It may be also observed that the accuracy is highest for LSTM–CNN–GS, and lowest for *K*-NN. For precision, the value for LSTM–CNN–GS is the highest, and lowest for *K*-NN. The Sensitivity value for *K*-NN is the lowest and highest for LSTM–CNN–GS. The specificity value is lowest for CNN but highest for LSTM–CNN–GS. The F-1 score is highest for LSTM–CNN–GS, and lowest for *K*-NN. The hyperparameter tuning method used in the study is grid search, and we observe that hyperparameter optimization has led to an increase in the overall performance of the model. Hence, the proposed model outperforms all the other baseline algorithms considered.

For dataset 2 (Table 4), we observe that the *K*-NN values for the evaluation parameters are 0.889, 0.896, 0.891, 0.918, and 0.889, respectively. For neural networks, the performance values are 0.892, 0.918, 0.911, 0.92, and 0.911, respectively,

whereas for CNN the values seem slightly better, i.e., 0.927, 0.918, 0.92, 0.927, and 0.92, respectively. Again, LSTM performs satisfactorily with values 0.915, 0.909, 0.92, 0.919, and 0.91. Another baseline model, CNN–LSTM demonstrates fair performance with values 0.938, 0.929, 0.936, 0.92, and 0.92, respectively. This performance improvement of CNN–LSTM may be attributed to the hybrid CNN–LSTM architecture. For LSTM–CNN architecture, output generated from the LSTM layer is fed into the CNN. The performance values for LSTM–CNN are better as compared to CNN–LSTM with values 0.947, 0.931, 0.922, 0.94, and 0.959. As mentioned earlier, the proposed model, LSTM–CNN–GS is an LSTM–CNN model with hyperparameter tuning using grid search. We observe that the performance of LSTM–CNN–GS has outperformed all the other baseline models with values 0.978, 0.982, 0.989, 0.99, and 0.972, respectively. It may be also observed that the values for accuracy, precision, sensitivity, specificity, and F-1 score are highest for LSTM–CNN–GS and lowest for *K*-NN. The hyperparameter tuning method used in the study is grid search, and we observe that hyperparameter optimization has led to an increase in the overall performance of the model. Hence, the proposed model outperforms all the other baseline algorithms considered.

The following figures represent the model performance across Dataset 1 and Dataset 2 with respect to the baseline algorithms and the proposed LSTM–CNN–GS-based deep neural network.

Figure 6a shows the accuracy values of all the models for both datasets. We observe that the accuracy for LSTM–CNN–GS for both datasets is higher than other benchmark models. Hence, hyperparameter tuning (grid search) increased our model performance, as accuracy values of LSTM–CNN–GS are better than LSTM–CNN.

Figure 6b shows the precision values of all the models for both datasets. We observe that the precision for LSTM–CNN–GS for both datasets is higher than other benchmark models. Hence, hyperparameter tuning (grid search) increased our model performance, as precision values of LSTM–CNN–GS are better than LSTM–CNN.

Figure 6c shows the sensitivity values of all the models for both datasets. We observe that the sensitivity for LSTM–CNN–GS for both datasets is higher than other benchmark models. Hence, hyperparameter tuning (grid search) increased our model performance, as sensitivity values of LSTM–CNN–GS are better than LSTM–CNN.

Figure 6d shows the specificity values of all the models for both datasets. We observe that specificity for LSTM–CNN–GS for both the datasets is higher than other benchmark models. Hence, hyperparameter tuning (grid search) increased our model performance, as specificity values of LSTM–CNN–GS are better than LSTM–CNN.

Figure 6e shows the F-1 score values of all the models for both datasets. We observe that F-1 score values for LSTM–CNN–GS for both the datasets are higher than other benchmark models. Hence, hyperparameter tuning (grid search) increased our model performance, as F-1 score values of LSTM–CNN–GS are better than LSTM-CNN.

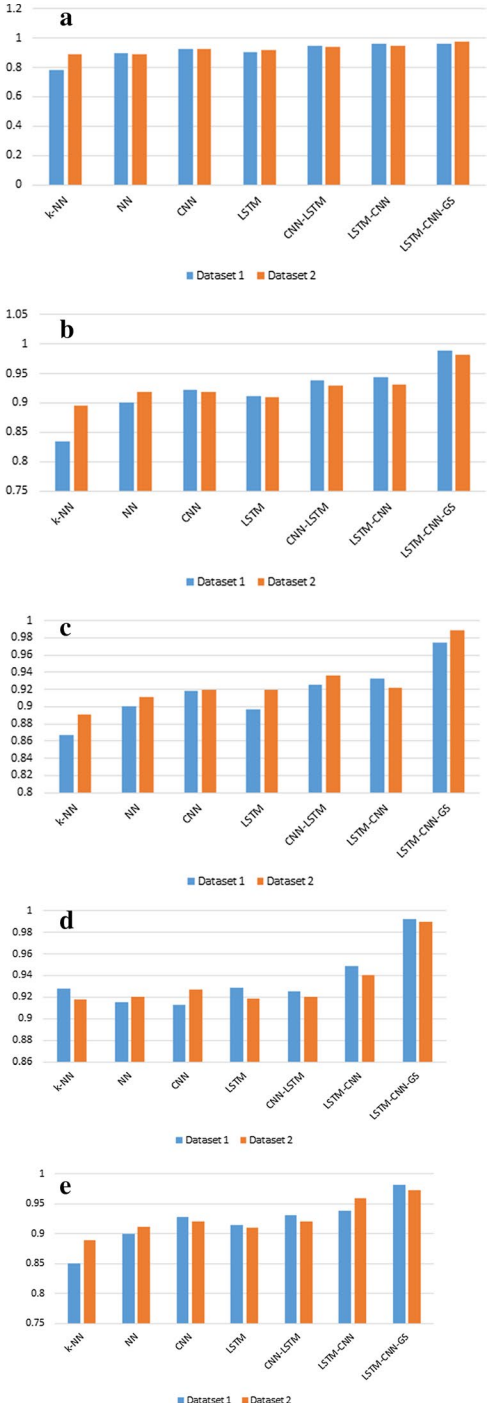

Fig. 6 a Model accuracy across datasets, **b** model precision across datasets, **c** model sensitivity across datasets, **d** model specificity across datasets, **e** model F-1 score across datasets

Table 5 Comparative analysis of our proposed work with previous works

References	Study/research	Methodology/parameters	Results
Das and Chakraborty [48]	Sentiment analysis (Amazon, IMDB)	TF-IDF and next word negation	Accuracy is 88.58% and 89.91%
Güner et al. [49]	Sentiment analysis (Amazon)	Machine learning techniques	LSTM performs the best with an accuracy of 90%
Bajeh et al. [50]	Performance analysis of particle swarm optimization	C4.5 decision tree, *K*-nearest neighbor and support vector machine	Accuracy < 71.23%, recall < 0.713
Iqbal et al. [51]	Hybrid framework for sentiment analysis	Genetic algorithm-based feature reduction	Accuracy for (IMDB < 76.7%, Amazon < 77.9%, Yelp < 75.3%)
Shreshtha and Nasoz [52]	Sentiment analysis (Amazon dataset)	Deep learning	Accuracy ~ 0.81, precision ~ 0.58, recall ~ 0.40
Rehman et al. [53]	Sentiment analysis (IMDB Movie Reviews)	Hybrid CNN–LSTM model	Accuracy is 91%
Tammina [54]	Sentiment classification in Telugu language	Hybrid learning (Lexicon based and machine learning)	Accuracy is 85%
Wang et al. [55]	Negative sentiment analysis during COVID-19 pandemic	Fine-tuned BERT (Bidirectional Encoder Representations from Transformers)	Accuracy is 75.65%
Dong et al. [56]	Sentiment analysis	Capsule network model with BiLSTM	Accuracy is 81.47%, 91.96%, and 48.34% for three datasets, respectively
Our proposed work	**Sentiment analysis**	**LSTM–CNN–GS-based deep neural network**	**For Dataset 1 and Dataset 2, accuracy is 0.964 and 0.978, precision is 0.989 and 0.982, sensitivity is 0.974 and 0.989, specificity is 0.992 and 0.99, F-1 score is 0.981 and 0.972**

4.2 Comparative analysis

In this section, we present a comparative analysis of our performed study with respect to some recent studies on sentiment analysis (Table 5).

Based on the experimental analysis and the comparative analysis, we observe that our proposed model, LSTM–CNN–GS performs relatively better than the other baseline algorithms on both datasets. Further, recent research works on sentiment analysis show that the performance of the model is acceptable, and that optimization has led to better model performance [57]. We make a few observations here, which are as follows:

- In this research, we propose a novel LSTM–CNN–GS-based deep neural network for sentiment analysis.
- We incorporate grid search as a hyperparameter optimization technique to minimize pre-defined losses and increase the accuracy of the model. Hence, hyperparameter optimization (grid search) led to an increase in the efficiency of the model (LSTM–CNN vs. LSTM–CNN–GS) (Fig. 6a–e).
- Our study reports better accuracy for both the datasets with respect to other baseline algorithms (Tables 3, 4).
- Our study reports the better performance of the proposed model with respect to many recent works on sentiment analysis (Table 5).
- Because the English language is diverse and incorporates infinite grammatical variations, misspellings, slang, complicated structures, it is not surprising that there may have been some misclassifications.

5 Conclusion

Sentiment analysis is one of the most popular research areas of natural language processing, which finds its use in various applications that may support smart cities. Past research works highlight sentiment analysis being performed using several conventional and non-conventional methods, including artificial intelligence. In this paper, we propose a novel LSTM–CNN–grid search-based deep neural network for the same. Baseline algorithms like convolutional neural networks, long short-term memory (LSTM), neural networks (NN), *K*-nearest neighbor (*K*-NN), and CNN–LSTM have also been considered for the study using multiple datasets. The parameters taken into account for evaluation are accuracy, precision, specificity, sensitivity, and F-1 score. Our study shows that the proposed model outperforms all the other baseline algorithms with an accuracy greater than 96%. The study also shows that hyperparameter tuning leads to an increase in model performance. In the future, we would like to explore the problem using more such hybrid machine learning techniques using other hyperparameter optimization methods. Also, the dataset considered consisted of English words primarily. It would be interesting to conduct the analysis on a dataset that has sentences from a different language. Since achieving 100% accuracy is a challenge (due to the diversity in the English language), the

speculation proposed about Sentiment analysis being a factor to distinguish humans and bots may be worth exploring.

Acknowledgements The authors would like to thank the Editor-inChief and the anonymous reviewers for their valuable comments and suggestions.

Funding This research received no external funding.

Declaration

Conflict of interest The authors declare no conflict of interest.

References

1. Lin C, He Y (2009) Joint sentiment/topic model for sentiment analysis. In: Proceedings of the 18th ACM Conference on Information and Knowledge Management, pp 375–384
2. Godbole N, Srinivasaiah M, Skiena S (2007) Large-scale sentiment analysis for news and blogs. Icwsm 7(21):219–222
3. Pasupathi S, Shanmuganathan V, Madasamy K, Yesudhas HR, Kim M (2021) Trend analysis using agglomerative hierarchical clustering approach for time series big data. J Supercomput 1–20
4. Hutto CJ, Gilbert E (2014) Vader: a parsimonious rule-based model for sentiment analysis of social media text. In: Eighth International AAAI Conference on Weblogs and Social Media
5. Gopi AP, Jyothi RNS, Narayana VL, Sandeep KS (2020) Classification of tweets data based on polarity using improved RBF kernel of SVM. Int J Inf Technol 1–16
6. Gräbner D, Zanker M, Fliedl G, Fuchs M (2012) Classification of customer reviews based on sentiment analysis. In: ENTER, pp 460–470
7. Ahmed KB, Radenski A, Bouhorma M, Ahmed MB (2016) Sentiment analysis for smart cities: state of the art and opportunities. In: Proceedings on the International Conference on Internet Computing (ICOMP). The Steering Committee of the World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), p 55
8. Priyadarshini I, Cotton C (2020) Intelligence in cyberspace: the road to cyber singularity. J Exp Theor Artif Intell 1–35
9. Priyadarshini I (2018) Features and architecture of the modern cyber range: a qualitative analysis and survey (Doctoral dissertation, University of Delaware)
10. Li W, Zhu L, Shi Y, Guo K, Zheng Y (2020) User reviews: sentiment analysis using Lexicon integrated two-channel CNN–LSTM family models. Appl Soft Comput 94:106435
11. Kumar RS, Devaraj AFS, Rajeswari M, Julie EG, Robinson YH, Shanmuganathan V (2021) Exploration of sentiment analysis and legitimate artistry for opinion mining. Multimed Tools Appl 1–16
12. Priyadarshini I, Wang H, Cotton C (2019) Some cyberpsychology techniques to distinguish humans and bots for authentication. In: Proceedings of the Future Technologies Conference. Springer, Cham, pp 306–323
13. Priyadarshini I, Cotton C (2019) Internet memes: a novel approach to distinguish humans and bots for authentication. In: Proceedings of the Future Technologies Conference. Springer, Cham, pp 204–222
14. Bukhari M, Bajwa KB, Gillani S, Maqsood M, Durrani MY, Mehmood I, Ugail H, Rho S (2020) An efficient gait recognition method for known and unknown covariate conditions. IEEE Access 9:6565–6477
15. Liu R, Shi Y, Ji C, Jia M (2019) A survey of sentiment analysis based on transfer learning. IEEE Access 7:85401–85412
16. Vo T, Sharma R, Kumar R, Son, LH, Pham BT, Tien Bui D, Priyadarshini I, Sarkar M, Le T (2020) Crime rate detection using social media of different crime locations and Twitter part-of-speech tagger with Brown clustering. J Intell Fuzzy Syst (Preprint), 1–13

17. Basiri ME, Nemati S, Abdar M, Cambria E, Acharya UR (2020) ABCDM: an attention-based bidirectional CNN–RNN deep model for sentiment analysis. Future Gener Comput Syst 115:279–294
18. Jin Z, Zhao X, Liu Y (2021) Heterogeneous graph network embedding for sentiment analysis on social media. Cogn Comput 13(1):81–95
19. Pota M, Ventura M, Catelli R, Esposito M (2021) An effective BERT-based pipeline for Twitter sentiment analysis: a case study in Italian. Sensors 21(1):133
20. Lu Q, Zhu Z, Zhang G, Kang S, Liu P (2021) Aspect-gated graph convolutional networks for aspect-based sentiment analysis. Appl Intell 1–12
21. Nemes L, Kiss A (2021) Social media sentiment analysis based on COVID-19. J Inf Telecommun 5(1):1–15
22. Priyadarshini I, Mohanty P, Kumar R, Son LH, Chau HTM, Nhu VH, Tien Bui D (2020) Analysis of outbreak and global impacts of the COVID-19. In: Healthcare, vol 8(2). Multidisciplinary Digital Publishing Institute, p 148
23. Dansana D, Kumar R, Das Adhikari J, Mohapatra M, Sharma R, Priyadarshini I, Le DN (2020) Global forecasting confirmed and fatal cases of covid-19 outbreak using autoregressive integrated moving average model. Front Public Health 8:580327
24. Tubishat M, Idris N, Abushariah M (2021) Explicit aspects extraction in sentiment analysis using optimal rules combination. Future Gener Comput Syst 114:448–480
25. Kandasamy I, Vasantha WB, Obbineni JM, Smarandache F (2020) Sentiment analysis of Tweets using refined neutrosophic sets. Comput Ind 115:103180
26. Jha S, Kumar R, Priyadarshini I, Smarandache F, Long HV (2019) Neutrosophic image segmentation with dice coefficients. Measurement 134:762–772
27. Jha S, Kumar R, Chiclana F, Puri V, Priyadarshini I (2019) Neutrosophic approach for enhancing quality of signals. Multimed Tools Appl 1–32
28. Quek SG, Selvachandran G, Munir M, Mahmood T, Ullah K, Son LH, Thong PH, Kumar R, Priyadarshini I (2019) Multi-attribute multi-perception decision-making based on generalized T-spherical fuzzy weighted aggregation operators on neutrosophic sets. Mathematics 7(9):780
29. Huang F, Li X, Yuan C, Zhang S, Zhang J, Qiao S (2021) Attention-emotion-enhanced convolutional LSTM for sentiment analysis. IEEE Trans Neural Netw Learn Syst
30. Zhao N, Gao H, Wen X, Li H (2021) Combination of convolutional neural network and gated recurrent unit for aspect-based sentiment analysis. IEEE Access 9:15561–15569
31. Srividya K, Sowjanya AM (2021) NA-DLSTM–A neural attention based model for context aware Aspect-based sentiment analysis. Materials Today: Proceedings
32. Priyadarshini I, Puri V (2021) A convolutional neural network (CNN) based ensemble model for exoplanet detection. Earth Sci Inf 1–13
33. Ramamurthy M, Robinson YH, Vimal S, Suresh A (2020) Auto encoder based dimensionality reduction and classification using convolutional neural networks for hyperspectral images. Microprocess Microsyst 79:103280
34. Patro SGK, Mishra BK, Panda SK, Kumar R, Long HV, Taniar D, Priyadarshini I (2020) A hybrid action-related K-nearest neighbour (HAR-KNN) approach for recommendation systems. IEEE Access 8:90978–90991
35. Jha S, Kumar R, Abdel-Basset M, Priyadarshini I, Sharma R, Long HV (2019) Deep learning approach for software maintainability metrics prediction. IEEE Access 7:61840–61855
36. Haq IU, Ullah A, Khan SU, Khan N, Lee MY, Rho S, Baik SW (2021) Sequential learning-based energy consumption prediction model for residential and commercial sectors. Mathematics 9(6):605
37. Tuan TA, Long HV, Kumar R, Priyadarshini I, Son NTK (2019) Performance evaluation of Botnet DDoS attack detection using machine learning. Evol Intell 1–12
38. Puri V, Jha S, Kumar R, Priyadarshini I, Abdel-Basset M, Elhoseny M, Long HV (2019) A hybrid artificial intelligence and internet of things model for generation of renewable resource of energy. IEEE Access 7:111181–111191
39. Petmezas G, Haris K, Stefanopoulos L, Kilintzis V, Tzavelis A, Rogers JA, Katsaggelos AK, Maglaveras N (2021) Automated atrial fibrillation detection using a hybrid CNN–LSTM network on imbalanced ECG datasets. Biomed Signal Process Control 63:102194
40. Zhu F, Ye F, Fu Y, Liu Q, Shen B (2019) Electrocardiogram generation with a bidirectional LSTM–CNN generative adversarial network. Sci Rep 9(1):1–11
41. Feurer M, Hutter F (2019) Hyperparameter optimization. In: Automated machine learning. Springer, Cham, pp 3–33

42. Liu B (2020) Text sentiment analysis based on CBOW model and deep learning in big data environment. J Ambient Intell Humaniz Comput 11(2):451–458
43. Nguyen LTK, Chung HH, Tuliao KV, Lin TM (2020) Using XGBoost and skip-gram model to predict online review popularity. SAGE Open 10(4):2158244020983316
44. Bittlingmayer AB (2019) Amazon reviews for sentiment analysis (Version 7) [A few million Amazon reviews in fastText format]. Kaggle. https://www.kaggle.com/bittlingmayer/amazonreviews/metadata
45. Maas A, Daly RE, Pham PT, Huang D, Ng AY, Potts C (2011) Learning word vectors for sentiment analysis. In: Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies, pp 142–150
46. Priyadarshini I, Mohanty PR, Cotton C (2021) Analyzing some elements of technological singularity using regression methods. Comput Mater Contin 67(3):3229–3247
47. Pritam N, Khari M, Kumar R, Jha S, Priyadarshini I, Abdel-Basset M, Long HV (2019) Assessment of code smell for predicting class change proneness using machine learning. IEEE Access 7:37414–37425
48. Das B, Chakraborty S (2018) An improved text sentiment classification model using TF-IDF and next word negation. arXiv preprint arXiv:1806.06407
49. Guner L, Coyne E, Smit J (2019) Sentiment analysis for Amazon. com reviews, Big Data in Media Technology (DM2583) KTH Royal Institute of Technology, Stockholm
50. Bajeh AO, Funso BO, Usman-Hamza FE (2019) Performance analysis of particle swarm optimization for feature selection. FUOYE J Eng Technol 4(1):149–154
51. Iqbal F, Hashmi JM, Fung BC, Batool R, Khattak AM, Aleem S, Hung PC (2019) A hybrid framework for sentiment analysis using genetic algorithm based feature reduction. IEEE Access 7:14637–14652
52. Shrestha N, Nasoz F (2019) Deep learning sentiment analysis of amazon.com reviews and ratings. arXiv preprint arXiv:1904.04096
53. Rehman AU, Malik AK, Raza B, Ali W (2019) A hybrid CNN–LSTM model for improving accuracy of movie reviews sentiment analysis. Multimed Tools Appl 78(18):26597–26613
54. Tammina S (2020) A hybrid learning approach for sentiment classification in Telugu language. In: 2020 International Conference on Artificial Intelligence and Signal Processing (AISP). IEEE, pp 1–6
55. Wang T, Lu K, Chow KP, Zhu Q (2020) COVID-19 sensing: negative sentiment analysis on social media in China via Bert Model. IEEE Access 8:138162–138169
56. Dong Y, Fu Y, Wang L, Chen Y, Dong Y, Li J (2020) A sentiment analysis method of capsule network based on BiLSTM. IEEE Access 8:37014–37020
57. Rokbani N, Kumar R, Abraham A, Alimi AM, Long HV, Priyadarshini I (2020) Bi-heuristic ant colony optimization-based approaches for traveling salesman problem. Soft Comput 1–20