ORIGINAL RESEARCH
published: 29 June 2022
doi: 10.3389/fncom.2022.900885

A Framework for Text Classification Using Evolutionary Contiguous Convolutional Neural Network and Swarm Based Deep Neural Network

*Sunil Kumar Prabhakar[1], Harikumar Rajaguru[2], Kwangsub So[1] and Dong-Ok Won[1]**

[1] Department of Artificial Intelligence Convergence, Hallym University, Chuncheon, South Korea, [2] Department of ECE, Bannari Amman Institute of Technology, Sathyamangalam, India

To classify the texts accurately, many machine learning techniques have been utilized in the field of Natural Language Processing (NLP). For many pattern classification applications, great success has been obtained when implemented with deep learning models rather than using ordinary machine learning techniques. Understanding the complex models and their respective relationships within the data determines the success of such deep learning techniques. But analyzing the suitable deep learning methods, techniques, and architectures for text classification is a huge challenge for researchers. In this work, a Contiguous Convolutional Neural Network (CCNN) based on Differential Evolution (DE) is initially proposed and named as Evolutionary Contiguous Convolutional Neural Network (ECCNN) where the data instances of the input point are considered along with the contiguous data points in the dataset so that a deeper understanding is provided for the classification of the respective input, thereby boosting the performance of the deep learning model. Secondly, a swarm-based Deep Neural Network (DNN) utilizing Particle Swarm Optimization (PSO) with DNN is proposed for the classification of text, and it is named Swarm DNN. This model is validated on two datasets and the best results are obtained when implemented with the Swarm DNN model as it produced a high classification accuracy of 97.32% when tested on the BBC newsgroup text dataset and 87.99% when tested on 20 newsgroup text datasets. Similarly, when implemented with the ECCNN model, it produced a high classification accuracy of 97.11% when tested on the BBC newsgroup text dataset and 88.76% when tested on 20 newsgroup text datasets.

Keywords: natural language processing, Differential Evolution, Particle Swarm Optimization, Convolutional Neural Network, deep neural network

Edited by:
Tien-Loc Le,
Sejong University, South Korea

Reviewed by:
Jakub Nalepa,
Silesian University of Technology, Poland
Ramalingam Sakthivel,
Chungbuk National University, South Korea

Correspondence:
Dong-Ok Won
dongok.won@hallym.ac.kr

Received: *21 March 2022*
Accepted: *17 May 2022*
Published: *29 June 2022*

Citation:
Prabhakar SK, Rajaguru H, So K and Won D-O (2022) A Framework for Text Classification Using Evolutionary Contiguous Convolutional Neural Network and Swarm Based Deep Neural Network. Front. Comput. Neurosci. 16:900885. doi: 10.3389/fncom.2022.900885

INTRODUCTION

The amount of data to be processed is too large with the advances in the development of information technology (Zheng et al., 2016). Extracting useful information from a huge amount of data is quite important as it saves a lot of time and effort. Therefore, in the field of engineering and technology, effective ways to extract useful information by filtering useless information is an important area of research (Johnson and Zhang, 2015). The representation of information is also pretty diversified including sound, text, images, etc. as there is a rapid rise in the enormous amount of data generated (McCallum, 1998). Fewer network resources are required by text when compared with sound and images, and it is easy for upload and download purposes. Text is always a vital carrier of information as even other forms of data can be represented by it. Achieving the fruitful results of text processing is highly time-consuming and therefore obtaining the data and processing it with effective algorithms can pave a way for good results (Conneau et al., 2017). Therefore, to achieve a free human-machine interaction, text classification becomes a quite interesting and critical technology in the field of Artificial Intelligence (AI). The main intention of text classification is to assign a huge quantity of text to one or more required categories depending on the content and attributes of the documents (Santos and Gatti, 2014). Rule-based and statistical classification-based methods are two categories of text classification (Onan, 2016). More knowledge and rule bases are required in the rule-based classification techniques. The rule development and the subsequent process of updating them make it quite difficult and suitable for application in a specific field. Depending on the statistical knowledge, these statistical learning methods are present, and they establish the learning parameters of the respective data model by means of applying sample statistics and then on the training set it is calculated and subsequently the training of the classifiers is conducted. Leveraging text classification methods is quite an important aspect in the field of Natural Language Processing (NLP) and text mining. For the text classification system, a large number of statistical machine learning techniques are implemented (Chen K. et al., 2016). In many real-time applications, text classification problems have been widely studied and represented in literature (Altinel and Ganiz, 2018). As far as the text classification and categorization of documents are considered, it is usually split into feature extraction, dimension mitigation, selection of classifiers, and evaluations. There are four various levels of the scope of text classification systems, namely document level, paragraph level, sentence level, and sub-sentence level (Comite et al., 2003). The classification tasks employed for text classification include KNN (Han et al., 2001), SVM (Sun et al., 2009), tree-based classifiers (Murthy, 1998), graph CNN methods (Yao et al., 2019), and other recently proposed deep learning models proposed in Li et al. (2020). Some of the most important and classic works of text classification are reported as follows.

A comprehensive survey of text classification algorithms was elaborated by Aggarwal and Zhai (2012). The advent of machine learning in automated text categorization was proposed in detail by Sebastiani (2002). The proposal of the explicit and implicit syntactic features for text classification (Post and Bergsma, 2013), the bay of tricks for efficient text classification (Joulin et al., 2017), and multilingual text detection with nonlinear neural networks (Li et al., 2015) are some of the trivial works done in the field of text classification. The phrasal and clustered representation was evaluated on a text categorization task in Lewis (1992), and the fast logistic regression for text categorization with variable length N-grams was proposed in Ifrim et al. (2008). DE was used for finely adapting Naïve Bayesian Classifier (NBC) and used for text classification in Diab and El Hindi (2017). A multiple partially observed view for multilingual text categorization (Amini et al., 2009), an iterative deep neighborhood model for text classification (Liu et al., 2020), integrating bidirectional Long Term Short Memory (LSTM) with 2D max pooling for text classification (Zhou et al., 2016), Recurrent Neural Network (RNN) for text classification with multi-task learning (Liu et al., 2016), Recurrent CNN for text classification (Lai et al., 2015), and a character level convolutional network for text classification (Zhang et al., 2015) are some of the most famous deep learning works proposed in the literature. A ranking based deep learning representation for efficient text classification (Zheng et al., 2018), a hierarchical neural network document representation approach for text classification using three different models (Kowsari et al., 2017), a C-LSTM neural network for text classification (Zhou, 2015), and a neural attention model for leveraging contextual sentences for text classification (Yan, 2019) are again some of the works which help the research community to a great extent for further analysis. Based on the recognition of semantic topics, the Chinese texts were classified by Chen Y. W. et al. (2016). An iterative dual attention network for text sentiment classification (Zhu et al., 2020), an aspect level sentiment classification with an interactive model of target and context (Han et al., 2019), and the sentiment analysis using common sense and content information were proposed in Agarwal et al. (2015). The news text classification technique and simulation depending on the hybrid deep learning model was developed in Sun and Du (2021), and an attention-based BiLSTM fused with CNN and gating mechanism for Chinese long text classification was done in Deng et al. (2021). A heterogeneous classifier ensemble based on deep learning and word embedding for text classification was done in Kilimci and Akyokus (2018), and a hybrid CNN-RNN attention-based model for text classification called CRNN was developed in Guo et al. (2018). Hybrid embedding-based text representation for hierarchical multi-label text classification (Ma et al., 2022), a co-attention network with label embedding for text classification (Liu M. et al., 2021), and a bidirectional gated temporal combination with attention for text classification (Ren et al., 2021) are some of the recent works reported in the field of text classification. The label-based attention for hierarchical multi-label text classification network was developed in Zhang et al. (2022) and the hybrid optimization algorithms with feature selection were utilized for text classification in Thirumoorthy and Muneeswaran (2021). As far as security aspects in text classification are concerned, for LSTM-based text classification, the backdoor attacks were mitigated with the help of backdoor keyword identification as implemented in Chen and Dai (2021).

In this work, two deep learning techniques are proposed with the help of swarm intelligence techniques like DE and PSO, and it has been incorporated with CNN and DNN, and finally, the results have been analyzed. DE is a very famous population-based evolutionary algorithm that is utilized for solving multi-dimensional global optimization problems over continuous spaces; thus, in this paper, it has been combined with deep learning for the purpose of text classification. The main intention to use PSO in this work is because of its simple concept, computational efficiency, easy implementation, and robustness to control parameters. Thus, considering these factors, it has been implemented with deep learning for the purpose of text classification in this paper. A few relevant works involving PSO and DE with deep learning for text classification and other important tasks are discussed as follows. The application of PSO for hyper-parameter selection in deep neural networks was done elaborately by Lorenzo et al. (2017a). The hyper-parameter selection in deep neural networks using Parallel PSO too was discussed by Lorenzo et al. (2017b). A PSO based deep learning model for vehicle classification (Alhudhaif, 2022), image classification (Junior and Yen, 2019), hyper spectral image classification (Liu X. et al., 2021), and flash flood detection from satellite images (Tuyen et al., 2021) too was reported in the literature. A text feature selection using the PSO algorithm was implemented by Zahran and Kanaan (2009). A feature selection empowered by self-inertia weight adaptive PSO for text classification was reported by Asif et al. (2022). An enhanced textual data classification using the PSO algorithm was reported by Aro et al. (2020), and the application of DE for neural networks optimization was performed by Baioletti et al. (2020). The feature selection for text and image data using DE with support vector machines and Naïve Bayesian classifiers was done by Dixit and Bansal (2020), and DE was used for fine tuning Naïve Bayesian classifiers with its applications for text classification implemented by Diab and El Hindi (2017). The DE based hyperparameters tuned deep learning models for disease diagnosis and classification were done in Kaliyapillai and Krishnamurthy (2020). The DE based feature selection and classifier ensemble for named entity recognition was done by Sikdar et al. (2012) and evolutionary optimization of ensemble learning to determine sentiment polarity in an unbalanced multiclass corpus was done by Gracia-mendoza et al. (2020). The organization of the paper is as follows. The contribution of the first proposed framework is given in Section First Proposed Framework: ECCNN. and the contribution of the second proposed framework is given in Section Proposed framework 2: Swarm DNN. The results and discussion are elaborated in Section Results and Discussion followed by the conclusion in Section Conclusion and Future Work.

FIRST PROPOSED FRAMEWORK: ECCNN

The proposed framework utilizes the idea of both DE and CNN and is explained as follows.

Differential Evolution

One of the most famous evolutionary algorithms used widely is DE (Mohamed, 2015). For various optimization tasks such as image processing, signal processing, wireless networking, computer vision, semantic classification, etc., it has been used widely. The training parameters are initialized by DE such as population size S, individual dimension S_{par}, crossover probability CR, and a mutation scaling parameter M. In the starting state, the generation of a population $P$of size $S$and dimension S_{par} is done utilizing the following equation:

$$p_i = Low_i + rand\left(S, S_{par}\right) * \left(Upp_i - Low_i\right),$$
$$p_i \in P, i = 1, 2, \ldots, S \tag{1}$$

where Low represents the lower frontiers of the search space and Upprepresents the upper frontiers of the search space. In a particular interval [0,1], a random matrix is initiated, and the function utilized is $rand(.,.)$. A novel and contemporary individual w_i is created from the present parent individual p_i by means of using a mutation operator. The mutation operation is performed by the DE scheme as defined in Equation (2).

$$w_i^t = p_{r_1}^t + M * \left(p_{r2}^t - p_{r3}^t\right) \tag{2}$$

where $p_{r1}, p_{r2}$and p_{r3} are different individuals which are chosen randomly for the population at an iteration t. An offspring individual from w_i and p_i can be generated by the crossover operator as follows:

$$y_{ij}^t = \begin{cases} w_{ij}^t, if \;\; \gamma_j \le CRor \;\; \delta_i \\ p_{ij}^t, \quad otherwise \end{cases} \tag{3}$$

where the arbitrary value picked for the j^{th}decision variable is represented as γ_j. A random version variable index picked from $\left[1, S_{par}\right]$ is represented as δ_i. The fitness functions fit_{p_i},$fit_{y_i}$of both the parent individual and the offspring p_i and y_i, respectively, are done separately.

The best individual for both the parent individual p_i and the offspring y_i is selected based on the computation of the fitness function values and is represented as per the following equation:

$$p_i^{t+1} = \begin{cases} y_i^t, \;\; if \;\; fit_{y_i} \le fit_{p_i}, \\ p_i^t, \quad otherwise \end{cases} \tag{4}$$

Unless the stop condition is satisfied, the repetition of the previous steps is done. The best individual is returned, and the DE is stopped if the stop condition is satisfied, or else from the mutation phase, it can continue to start again. To perform a mutation, various strategies can be utilized by the DE algorithm where the capability of the search space in terms of exploration can be improved. By representing it "DE/c/d", the distinguishment between strategies can be done, where the solution to be mutated is denoted by 'c' and the number of varied solutions utilized is represented by 'd'. Two strategies are utilized in this work, where the first one is represented as "DE/best/1" and is represented as:

$$w_i^t = p_d^t + M * \left(p_{r2}^t - p_{r3}^t\right), \tag{5}$$

where the second one is represented as "DE/best/2" and is represented as:

$$w_i^t = p_d^t + M * \left(p_{r2}^t - p_{r3}^t\right) + M * \left(p_{r3}^t - p_{r4}^t\right) \tag{6}$$

where the best solution at iteration t is indicated by p_d^t

Proposed Deep Learning Model ECCNN

The deep learning model proposed is that it considers the input instances of a particular data point and at the same time it considers the classification map of contiguous data points too as the input. The inputs of the model are introduced followed by the model construction and the technique required to learn the model parameters are then explained.

Inputs to the Model

It is considered that a training set of n datapoints is present and a multiclass classification issue of l classes is present. The consideration of the training set is done as $\left\{\left(P_i, q_i\right)\right\}_{i=1}^{n}$, where the input data which presents the i^{th} data points are represented as P_i and the class label vector of the i^{th} data point is represented as $q_i = \left[q_{i1}, \ldots, q_{iL}\right]^T \in \{1, 0\}^L$. $q_{il} = 1$ if P_i belongs to the l^{th}class and it is zero otherwise. In the training set P_i, in order to classify one data point, two types of data are included by the input of the model as follows:

Instance Arrangement

The instances of the data point itself are the first type of input and are considered as follows:
$P_i = \left(p_{i1}, \ldots, p_{i|P_i|}\right)$, where the length of the sequence is expressed as $|P_i|$ and the feature vector of the k^{th} instance of the i^{th} data point is represented as p_{ik}.

Contiguous Classification Map

The neighborhood of P_i is the second type of input and the classification map of the neighborhood. The neighborhood dataset of P_i is specified as N_i. The classification response is initially computed for every class and then a class wise max-pooling operation is applied. The L maximum responses for L classes are finally concatenated to obtain the classification map of N_i. With respect to the l^{th} class, the classification response of a data point $x_j \in N_i$ is denoted as $x_{jl} \in [0, 1]$, and the classification map of N_i is expressed as follows:

$$x_i = \left[\max_{j:P_j \in N_i} x_{j1}, \ldots, \max_{j:P_j \in N_i} x_{jL}\right]^T \in [0, 1]^L \tag{7}$$

where $\max_{j:P_j \in N_i} x_{jl}$ denote the max pooling outcome of the classification acknowledgment over N_i with respect to the l^{th}class. The computation of the classification response x_{jl} is expressed in the following subsections.

Based on the above explanations, for every input data point P_i, 2 inputs are expressed as $\left(p_{i1}, \ldots, p_{i|P_i|}\right)$ and x_i.

Structure of the Model

Figure 1 illustrates the overview framework of our model. The model comprises a CNN model (Jeong et al., 2020), indicated as c, one concatenation layer, 1 Fully Connected (FC) layer, and one Softmax nonlinear transformation layer inspired by the work and architecture in (Liu et al., 2020). The dataflow in the model along with the function of these layers are explained as follows:

(i) The initial transformation of the input sequences of instances to a vector of a d-dimensional vector $y \in \Re^d$ is done by the CNN model $'c'$which has 3 convolutional layers and 2 max-pooling layers represented as:

$$y_i \in f\left(P_i\right) \tag{8}$$

(ii) The concatenation of y_i is done by utilizing the neighborhood classification map vector $x_i \in \Re^L$ by using the concatenation layer. The indication of the concatenated vector is done as follows:

$$\begin{bmatrix} y_i \\ x_i \end{bmatrix} \in \Re^{d+L} \tag{9}$$

(iii) The fully connected layer enables the concatenated vector to further mitigate it to an L-dimensional vector and its concatenation weight matrix is expressed as

$$W = [w_1, \ldots, w_L] \in \Re^{d+L} \times L \tag{10}$$

where w_l represents the l^{th} column which corresponds to the l^{th} class. The calculation of the output of the FC layer is expressed as

$$W^T \begin{bmatrix} y_i \\ x_i \end{bmatrix} = \left[W^T \begin{bmatrix} y_i \\ x_i \end{bmatrix}, \ldots, W_L^T \begin{bmatrix} y_i \\ x_i \end{bmatrix}\right]^T \tag{11}$$

(iv) The Softmax activation layer is used to enable the outputs of the FC layer to normalize the probabilities on the L classes and the calculation is done as follows:

$$\overline{q_i} = \left[x_{i1}, \ldots, p_{iL}\right]^T \in [0, 1\,]^L \tag{12}$$

where,

$$x_{il} = \frac{\exp\left(w_1^T \begin{bmatrix} y_i \\ x_i \end{bmatrix}\right)}{\sum_{l'=1}^{L} \exp\left(w_{l'}^T \begin{bmatrix} y_i \\ x_i \end{bmatrix}\right)} \tag{13}$$

denote the probability of P_i which belongs to the l^{th} class. The $\overline{q_i}$ is conveyed as the model output vector. The class with the largest probability is chosen in order to determine the class of the designated data point P_i and is expressed as

$$q^* = \arg\max_{l=1,\ldots,L} x_{il} \tag{14}$$

Parameter Learning of the Model

The parameter of the CNN model $'c'$, the contiguous classification mapping points x_i, and the connection weight matrix are the three groups of parameters in our proposed model. A unified learning framework is built in order to learn

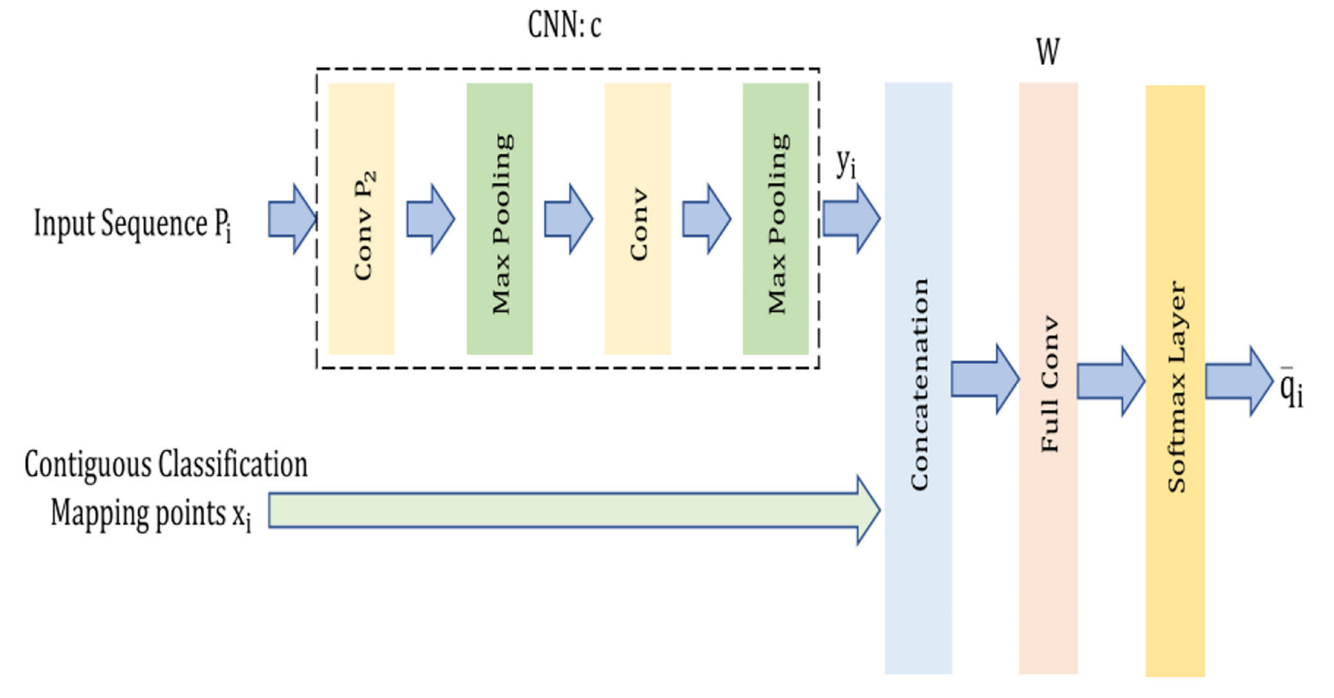

FIGURE 1 | Overview framework of the CCNN model.

the parameters, so the training data is fit well. By means of using cross-entropy loss functions, the classification error is measured, and by using the squared l_2 norm of the parameters, the model complexity is analyzed in this specific learning framework. To mitigate the overfitting risk, minimizing the classification error is proposed so that the classification performance can be improved, and the complexity of the proposed model can be reduced.

The modeling of the learning problem is done as a minimization problem and the problem objective is expressed as follows:

$$obj\,(W,c) = \sum_{i=1}^{n} l\left(q_i,\overline{q_i}\right) + \frac{T_1}{2}\left(\|W\|_2^2 + \|c\|_2^2\right) \tag{15}$$

where $l\left(q_i,\overline{q_i}\right) = -\sum_{l=1}^{L} q_{il}\log\left(\overline{q}_{il}\right)$denote the cross-entropy loss function for the i^{th} data point. The squared l_2 norm of $W$is represented as $\|W\|_2^2$. The squared l_2 norm of the filters of the CNN model c is represented as $\|f\|_2^2$ and T_1 denotes the tradeoff parameters of the l_2 norm regularization term and the classification error term. To grasp the quintessential parameters and c^*across the training set, the minimization problem is expressed as follows:

$$\begin{aligned} &W^*, c^* = \arg\min_{W,c} O\,(W,c)\,, \\ &s.t.y_i = f\,(P_i)\,, \forall_i = 1,\ldots,n \end{aligned} \tag{16}$$

In the learning problem, the convolutional specification vector of the CNN model y_i is introduced for every position of the data and it is inflicted to be identical to the CNN model output, $c\,(P_i)$.

Optimization Problem

Solving the problem in Equation (16) directly is quite hectic and the classification map x_i is fundamentally a function of W, c and $x_{j:P_j\in N_i}$. Also, there is a coupling of parameters $W$and c. Therefore, to solve these problems, the famous Expectation Maximization (EM) algorithm is utilized in this work (Wang et al., 2004). In an alternate manner, updating the parameter specification and the contiguous classification map vectors for every data point is done in an iterative algorithm. The classification map vectors $x_i|_{i=1}^{n}$ are fixed in M-step and then by minimizing the objective, $W$and c are updated. The fixing of the parameters $W$and c are done in the E-step so that the neighborhood and the contiguous classification map vectors are updated.

Proposed Framework Implementation of the ECCNN

The proposed ECCNN framework dependent on the DE algorithm for the enhancement of CCNN is explained in detail. To estimate the optimal architectures and parameters for a CNN, ECCNN is proposed so that the performance of classification is enhanced. The best configuration is searched by DE from a group of parameters and is utilized to assemble and train a CNN. The surviving CNN architectures which are utilized for the classification of text in the form of a word, sentence, etc., usually have just 1D convolution and pooling in most cases;

however, 2D convolution operators are implemented in ECCNN. The text from the dataset is provided as an input to the CNN in the form of a word insertion matrix with two dimensions. Therefore, the representation of every word is done by the vector extricated from a word insertion or fixing which is pretrained. To inhibit destroying the word embedding structure, the extraction of features is aided by the 2D convolution operators. The ECCNN consists of three stages, namely starting stage, evaluation stage, and updating stage. A random population $P$with size $S$and dimension $S_{par}$is created in the starting stage by ECCNN. The total number of hyperparameters utilized to dominate and regulate the CNN configuration is represented as S_{par}. The CNN configuration is nothing but the convolution filters, dropout rate, filter size, convolution filter size, the total number of filters, and the number of neurons in the FC layer. A random value is selected and utilized to start every solution P_i, $(i = 1, 2, \ldots, S)$ and various values are contained in every parameter in P.

The division of the dataset into training/testing with a 10-fold method is utilized by the two fitness evaluation techniques. Secondly, based on the present P_i, the CNN is built by the evaluation stage, where the determination of the number of parallel convolution layers is done by the number of convolution filters. To a parallel convolution layer, the convolution filter size is assigned, and it is followed by a max-over time pooling layer, which helps to mitigate the dimensionality and computational cost. The representation of pooling size is done as $\left(\max(L) - fy + 1, 1\right)$, where the sequence length is represented by L and the filter size which has allocated the previous convolution layer is expressed as fy. In order to consolidate the output from every pooling layer, a concatenation operation is performed, and then it is fed to the FC layer. Depending on the corresponding values from P_i, a hidden layer followed and accompanied by a drop out functioning is built by the ECCNN. Once the CCNN is built using P_i, for the performance enhancement of the CNN model, the testing set is utilized, and it is done by means of utilizing the fitness value function fit_i of the ongoing solution P_i. Then the best solution P_{best} is selected by DE which has the highest fitness values fit_{best}. Lastly, using the updating stage, the updation of the potential and final solutions of the population $P$is obtained utilizing DE algorithm operators which involve cross over, mutation, and selection steps. Unless the stop criterion is met, the repetition and evaluation of updation stages are done. **Figure 2** gives the detailed explanation of the proposed model. The following section gives more comprehensive details about it.

Initializing Stage

The generation of the catalog of values that corresponds to every parameter is done. Moreover, the setting of the parameters of the DE algorithm is also done. Choosing the size of the solutions S along the utmost number of iterations $t_{\max}$ is also given a higher significance in this process. The generation of a random integer population P is expressed in the following equation with reference to a particular size and dimension S_{par} as follows:

$$p_{ij} = low_j + rand * \left(upp_j - low_j\right), \quad j = 1, 2, \ldots, S_{par}, i = 1, 2, \ldots, S \tag{17}$$

The lower and upper frontiers of the j^{th}parameter of $p_i \in P$are represented as L_j and U_j, respectively.

About 500 perceptible filters with similar sizes and various initialization values are present as far as the size of the convolution filter is concerned. After every convolution layer, the pooling layer along with the max-pooling operation is accommodated which assumes a similar filter size value so that the computation of the pooling size is done quickly. From the pooling layer, the output feature vectors are assembled along with the contiguous classification mapping points, and then a concatenation operation is implemented so that it can be implemented directly into the FC layer. About 400 neurons are comprised in this FC layer and it is initialized by utilizing the uniform mode. In order to get a high classification accuracy at the end of the model, the Softmax layer is utilized.

Evaluating Stage

The construction of the CNN model is done at this stage depending on the parameters of the current solution p_i. A 10-fold split technique is utilized to assess the fitness function fit_i for every P_i. 90% of training data is selected as the training set and 10% of data is selected as the testing set. For 10 times, this evaluation is repeated and the fitness function value fit_i is nothing but the classification accuracy average over ten times and is represented as

$$fit_i = \frac{\sum_{l=1}^{10} acc_l}{10} \tag{18}$$

where acc_l specifies the classification accuracy for the l^{th} run.

Updating Stage

The determination of the finest solution P_{best} with the highest fitness function value fit_{best} is done in this stage. The updation of every solution P_i in the current population $P$is done by means of utilizing the three main operators of the DE algorithm. Unless the stop criterion is satisfied, the repetition of the evaluation stage and the update stages are done continuously. When the maximum number of iterations ($t_{\max}$) is reached, it implies that the stop condition is achieved.

Impact of CNN Parameters on the Architecture

The performance of the simulation results is affected greatly by the parameter tuning for all the deep learning algorithms. The developed ECCNN architecture with suitable parameters is explained as follows. An individual configuration is required in order to parametrize the CNN model. The parameters of the individual structure from two layers, namely the convolution layer and the FC layer, are considered in our experiment. The convolution layers are trained at the starting and in total the coding of five various parameters is done for each individual. For every individual, the optimizer is fixed, and the operation is merged. To enhance the accuracy of the classified sentences to the best of their ability on a test set, the parameter values are also changed. The parameters and their respective values are expressed in **Table 1**.

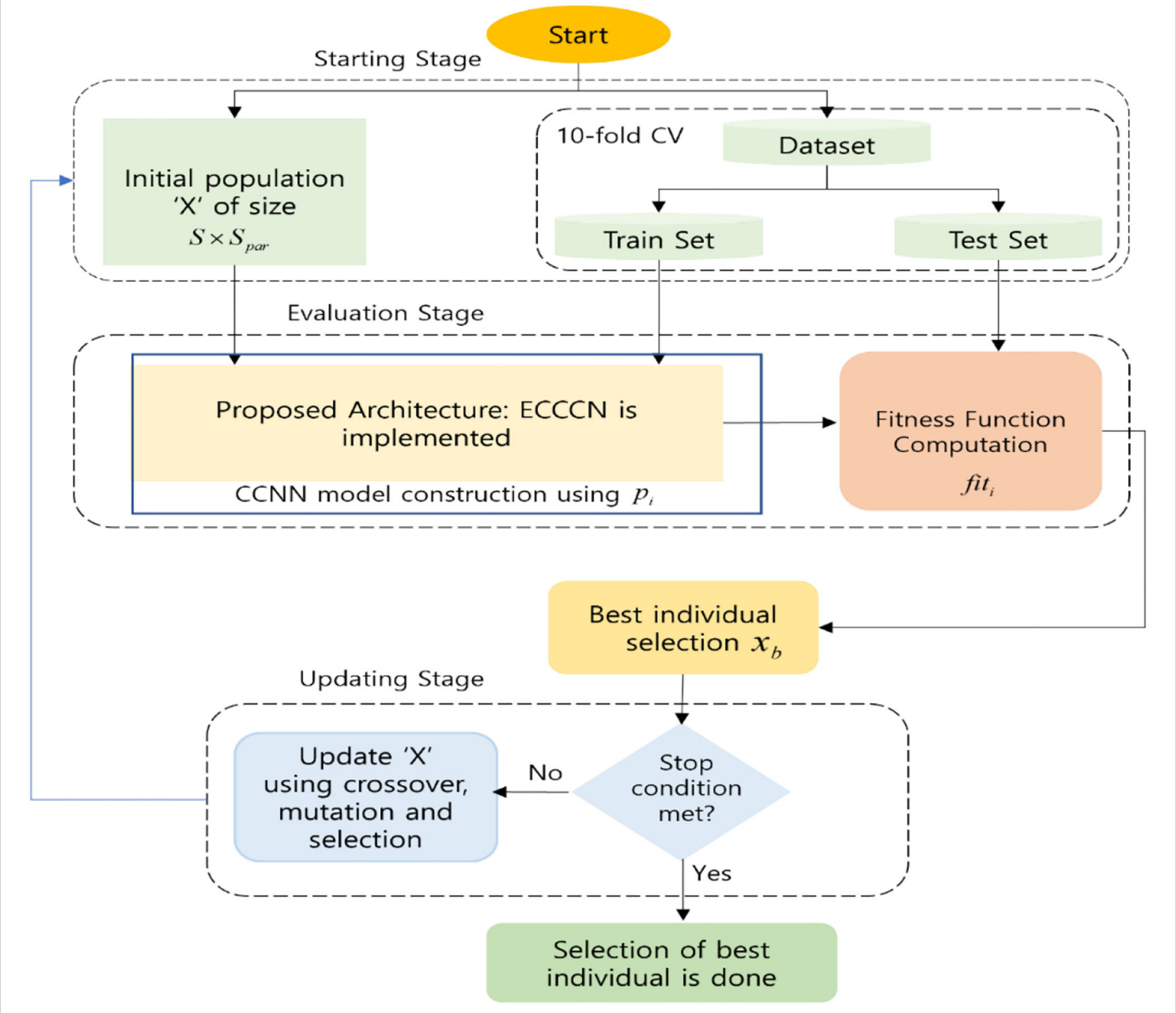
Start
Starting Stage
Initial population 'X' of size $S \times S_{par}$
10-fold CV
Dataset
Train Set
Test Set
Evaluation Stage
Proposed Architecture: ECCCN is implemented
CCNN model construction using p_i
Fitness Function Computation fit_i
Best individual selection x_b
Updating Stage
Update 'X' using crossover, mutation and selection
No
Stop condition met?
Yes
Selection of best individual is done

FIGURE 2 | Framework of the proposed ECCNN.

TABLE 1 | Parameter Values of the proposed ECCNN architecture.

Parameter	Value
Specifications of Convolution Layer:	
Size of the filters	2 to 10
Number of filters per convolution filter size (NFCS)	50, 100, 150, 200, 250, 300, 350, 400, 450, 500
Specification of Fully Connected Layer:	
Dropout Rate Setting	0.2 to 0.8
Initialization Mode Setting	Uniform, normal, LeCun uniform, He uniform
Neuron Number	50, 100, 200, 250, 300, 350, 400

To perform the convolution operation, the number of filters utilized is in the range of 50 to 500 filters per filter size. The contiguous classification mapping points along with the total number of convolution layers are trained in a collateral manner and are associated with the dimension of filter size. To generate a random filter size list, a random function is incorporated where every filter size has values ranging from 2 to 10. In a list, the highest number of generated filter sizes is restricted to only a few ones. To construct the FC layer, the number of neurons utilized is 50, 100, 200, 250, 300, 350, and 400. For the convolution and FC layer, the activation function utilized is the ReLU function. For the FC layer, various initialization nodes are explored such as normal, He uniform, LeCun uniform, and

TABLE 2 | Architecture specification implementation for the 2 datasets using ECCNN architecture.

Dataset utilized	Filter size	Total number of neurons considered	NFCS	Initialization mode	Dropout rate
20 Newsgroup	[2,3,5]	300	100	Normal	0.5
BBC news group	[2,3,5]	300	100	Normal	0.4

uniform. The famous regularization technique Dropout rate is adopted to avoid the overfitting of CNN and the dropout rate has different ranges starting from 0.2 to 0.8. To train the CNN, the optimizer utilized is Adam. Zero padding is adopted so that sentences with variable lengths can be handled well. To choose the paradigmatic arrangement for every dataset, a 10-fold cross-validation has been utilized to compute the final classification result. The training of the CNN will be done for the dataset from scratch using the selected arrangement and configuration. Thus, for the proposed ECCNN architecture, after several trials and error methods, the specifications which provided good results are analyzed and provided in **Table 2** with the subsequent DE parameters such as population size = 60 or 90; DE strategy = DE/best/1, and DE/best/2; and for all the varying values of *M*and *CR*, the results are tabulated in **Tables 5–10**. The population size was set as 60 and 90 after several combinations of trial and error, and it was set to those values as it finally gave a good result.

PROPOSED FRAMEWORK 2: SWARM DNN

The proposed framework makes use of the idea of both PSO and DNN and it is implemented as follows.

Hyperparameter Selection of DNN

A PSO-dependent algorithm (Kennedy and Eberhart, 1995) is utilized in order to select the hyperparameters of DNN. To construct the DNN, this algorithm is highly useful. A DNN is nothing but a multilayer ANN with many hidden layers (Schmidhuber, 2015). There is a full connection among the weights of DNN. The connection of every neuron in a distinct layer to all neurons of the higher end layer is situated next to it and is well established. In a feedforward manner, the propagation of the information in DNN is done (i.e.,), from the input layer to the output layer through the hidden layer. The typical DNN structure is depicted in the following **Figure 3**.

For different machine learning tasks, DNNs are used widely. In terms of performance, it has surpassed the ability of most of the machine learning techniques. Selecting the hyperparameter value is quite important for the performance enhancement of the DNN. For the given machine learning task, the performance of DNN is influenced by the DNN hyperparameters which help to implement it as a collection of important values that checks the behavior, architecture plan, performance accomplishment, etc. The two kinds of hyperparameters utilized are layer-based parameters and global parameters. The general behavior of DNN is defined by global parameters such as batch size, learning rate, number of layers, epochs number, and the utilized optimizer. For each layer in DNN, there is a huge dependence on layer-based parameters. Typical examples of layer-based parameters include the number of neurons, type of layer, activation function, and the initialization method. Hyperparameters can always differ from one assignment to another assignment and therefore they must be adjusted and regulated before the training process. To tune the DNN hyperparameters precisely, a machine learning expert who is specialized and updated with the current pattern recognition tasks can be utilized to overcome this problem, but the presence of such an expert is not available in most cases. The trial and error method can also be used as a potential solution to adjust these hyperparameters in a manual manner. By implementing random search/grid search, the execution of hyperparameters is done so that the search space can be handled well. On the defined attribute of hyperparameters, a grid search is executed where the identification of those specific ranges depends on the preceding proficiency of the underlying task. Then the hyperparameter values are picked up from the predetermined assortment incessantly and on the training set, the performance of DNN is evaluated. When the testing of all the amalgamation of hyperparameter values is done, the selection of the finest combination is obtained, and it is utilized to configure the DNN and enables it to evaluate it on the testing set. Grid random search and random search are almost similar to each other, but in a random search, the hyperparameter values are selected in a methodical manner and then the hyperparameter values are selected by the user from these predefined ranges in a random fashion. Bayesian optimization seems to have been a good technique for the selection of hyperparameters, however, the complex nature and huge search space criteria of the hyperparameters value of DNN make such manual algorithms exhausting. To trace the global optima of a nonlinear function, a metaheuristic algorithm/evolution is utilized which acts in a wonderful manner. To solve the DNN parametrization problem in an automatic manner, evolutionary algorithms (EA) seem to be a very versatile and promising technique. Therefore, utilizing EA for DNN hyperparameter optimization to achieve a high classification accuracy is favored.

PSO

In a continuous search space, for the optimization of non-linear functions, the famous metaheuristic algorithm utilized is PSO (Kennedy and Eberhart, 1995). The social behavior of animals is mimicked by PSO and has a collection of many members called particles. The swarm size is nothing but the total number of particles in the swarm denoted by *S* and is an integer value. Two vectors of *N* lengths are present in each particle of the specific swarm, where the size of the dimension is expressed as

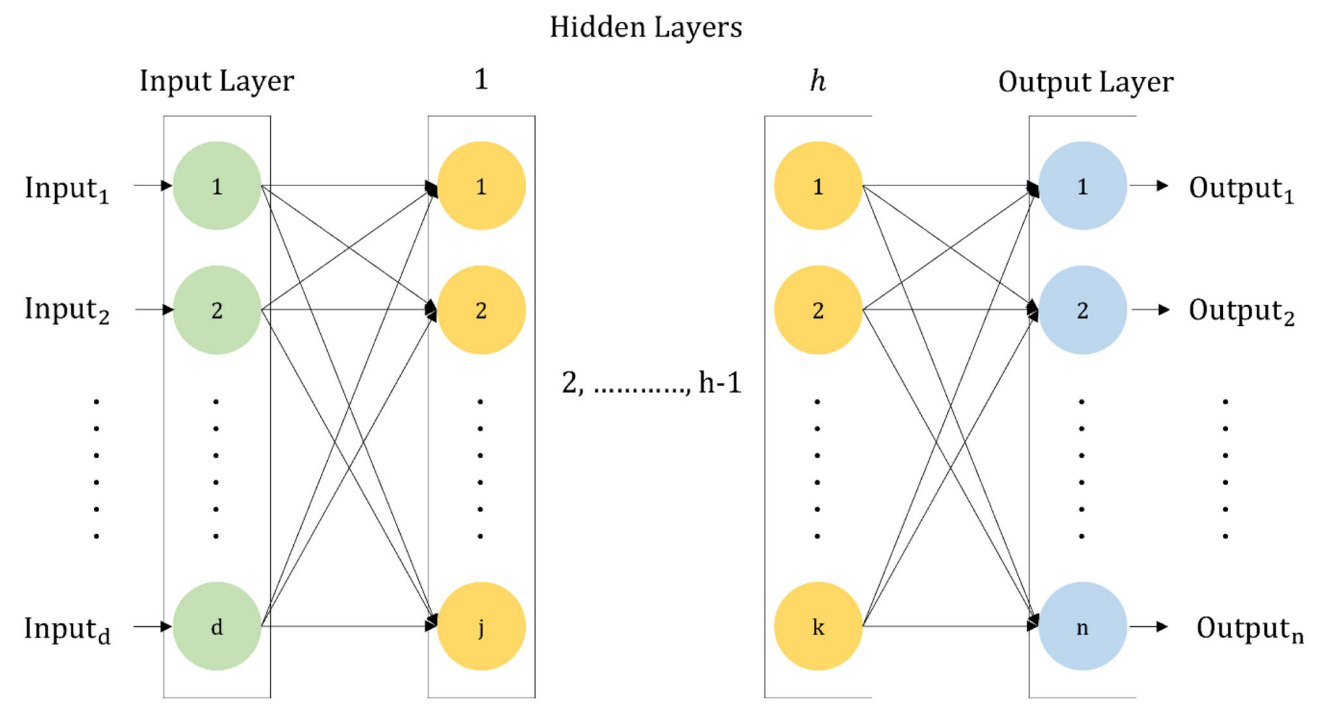

FIGURE 3 | A standard structure of DNN.

N. In the search space of the problem, the identification of the current position of the particle is done and it is called the first vector/position vector indicated by P. The candidate solution to the specific problem here is nothing but the indication of the position vector. During the next iteration, in the search space of the problem, both the speed and direction of the particle are determined by the second vector termed velocity vector, which is indicated by V. The storing of another two vectors is quite important in the execution of PSO during every iteration. The personal best vector is the initial vector and indicated by P^i_{best} which specifies the best position of the i^{th}particle in the swarm. During every iteration, the updation of the independent personal best vector for every particle in the swarm is done. The best position traced in the swarm so far is indicated by the secondary vector known as the global best vector and is specified by G_{best}. In the swarm, for all the particles, an exclusive global best vector is found, and it is updated at every iteration. The collective knowledge of the swarm is represented by the global best vector; however, the personal best vector is used to represent the cognitive knowledge of the particle. In the swarm S, for every particle i at iteration t, the updation of the velocity $V$and particle P vectors to the next iteration $t + 1$is expressed as

$$V^i_{t+1} = WV^i_t + C_1 r_1(t) \left(P^i_{best} - P^i_t\right) + C_2 r_2(t) \left(G_{best} - P^i_t\right) \quad (19)$$

$$P^i_{t+1} = P^i_t + V^i_{t+1} \quad (20)$$

The influence of the particle velocity at the present iteration to the successive iteration is controlled by the inertia weight constant W. It is done so that the particle does not deviate outward to the problem search space by adjusting the particle speed and direction. The acceleration coefficients are expressed as C_1 and C_2 are constants. The random values are r_1 and r_2, and are uniformly distributed in the ranges of [0,1]. The random computation of the new values r_1 and r_2 is done at the start of every iteration and they are usually constant for all the swarm particles of that particular iteration. To scale both the cognitive knowledge and the collective knowledge of the particle swarm, there is a change in velocity parameter, and the constants used for this process are C_1, C_2, r_1, r_2. Thus, the optimal solution to the problem is obtained by means of new position vectors of all the particles.

PSO for the Solution of DNN Hyperparameters

As an optimization task, the interpretation of the selection of hyperparameters of DNN is done. Minimizing the loss function $L(D, T)$is the primary objective, where the training set is indicated by $T$and the DNN model is indicated by D. The vector of the optimized hyperparameters $H$is given as an output by the PSO optimization algorithm so that the loss function is minimized after the contribution of the DNN model D. Then the hyperparameters H are utilized to trace it and then it is trained on the training set T. The function $F^* : R^N \rightarrow R$ is nothing but a fitness function of the PSO-based technique, where the mapping and the training of DNN are done from a real-valued vector with a hyperparameter of length N to the accuracy of the DNN which has a real value. The hyperparameter vectors are used to trace it and finally on the test set Y, it is tested. Among all the combinations of hyperparameters, the optimal hyperparameter vector is found by the PSO algorithm so that maximum accuracy is obtained when the trained DNN is tested on the test set. The selection of hyperparameters can be any by the user to make sure that the generality of the PSO algorithm is dependent on DNN. Therefore, the hyperparameters number and the domain related to each parameter which can be set to all possible values can be well assigned by the user. The PSO-based algorithm is so efficient that all the hyperparameter vectors indicating the particles are initialized in the swarm.

For predefined parameter ranges, the validation of the proposed algorithm is quite important during execution so that the updated position and velocity vectors are authenticated. At the end of every iteration, the simultaneous checking of two various stop conditions is done so that the computations are reduced, and the convergence is made faster. The primary condition transpires when the outcome of the global best vector in terms of fitness becomes fewer than a particular threshold value ε. When the paramount number of iterations is not extended to reach the threshold value and there is no improvement in the global best vector, then this case occurs. If the paramount number of iterations are carried out well, then the second condition happens. The optimal solution H provides the global best vector when either the primary/secondary condition is contented and then the search process is terminated.

The four important steps are as follows:

Inputs: A training set $T$and Test set Y, Swarm Size S, Acceleration constant C_1, C_2, Number of hyperparameters N, evolution threshold parameter(ε), Inertia constant W, paramount number of iterations ($t_{\max}$), minimum and maximum velocity value ($V_{\min}$) and ($V_{\max}$). The final optimal solution is given as H

Step 1: Preprocessing Phase:

a) The process is initialized
b) The inputs $N, S, V_{\min}, V_{\max}$ are given
c) The domains for $h^k, k \leftarrow 1$ to $N$are defined.
d) The hyperparameters and velocity generators are created
e) The P and Vvectors of S particles are initialized each of N length.

Step 2: Initialize Phase:

a) Input $T, Y, \varepsilon, C_1, C_2, W, t_{\max}$
b) $P^i_{best} \leftarrow -\infty, i \leftarrow 1 to S$
c) $G_{best} \leftarrow -\infty$
d) For all Sparticles, $F * (P)$ is computed and then P^i_{best} is updated.
e) G_{best} is updated
f) $t \leftarrow 1$

Step 3: Evolution Phase:

a) $r_1(t)$ and $r_2(t)$ are computed
b) $V, P, F * (P)$and P^i_{best} for all the Sparticles are computed
c) G_{best} is updated
d) $t \leftarrow t + 1$
e) Check for stop conditions. If yes, proceed to the finish stage and if no, repeat the evolution stage

TABLE 3 | PSO parameters values/ranges.

Parameter	Value/Ranges
C_1	0 to 4
C_2	0 to 4
S	[5, 75]
$V_{\min}$	0
$V_{\max}$	1
W	[0.5, 0.8]
$t_{\max}$	[20, 120]
ε	[0.1, 0.00001]

Step 4: Finishing Phase:

a) $H \leftarrow G_{best}$ is given as output
b) Terminated

Parameters of PSO

The PSO parameter $(S, C_1, C_2, W, V_{\max}, V_{\min}, t_{\max}, \varepsilon)$selection is quite a complex process. Previous studies have recommended quite a lot of values of the PSO parameters that could be utilized. The value ranges for every PSO parameter are shown in **Table 3**. For each parameter, a value is selected randomly and during the PSO execution, it is fixed as a constant.

Implementation of Swarm DNN

The description of the proposed Swarm DNN in text classification is as follows. The proposed methodology comprises four successive stages such as (a) Initialization Stage, (b) Optimization Stage, (c) Extraction of meaningful results Stage, and finally (d) Termination Stage. All the necessary operating parameters are initialized, and the text input files are prepared well. The text comprises the training set which is followed and accompanied by the test set. The parameters for PSO elements in DNN experiments are set as described in the four cases. The experiment was repeated for hundreds of cases by trial-and-error method and finally, four cases that gave the highest classification accuracy results were considered and analyzed in this work.

Case 1:

$$S = 15, V_{\min} = 0, V_{\min} = 1, t_{\max} = 60, \varepsilon = 10^{-2},\\ C_1 = C_2 = 1,\ W = 0.5$$

Case 2:

$$S = 30, V_{\min} = 0,\ V_{\min} = 1, t_{\max} = 80,\\ \varepsilon = 10^{-3}, C_1 = C_2 = 1, W = 0.6$$

Case 3:

$$S = 50, V_{\min} = 0,\ V_{\min} = 1, t_{\max} = 100,\\ \varepsilon = 10^{-4}, C_1 = C_2 = 2, W = 0.7$$

Case 4:

$$S = 65,\ V_{\min} = 1, t_{\max} = 120,\\ \varepsilon = 10^{-4}, C_1 = C_2 = 2, W = 0.8$$

TABLE 4 | DNN hyperparameter and its domain.

Hyperparameters	Domain	Explanation
Momentum	[0.1, 0.9]	Continuous
Learning Rate	[0.1, 0.9]	Continuous
Drop rate	[0.1, 0.9]	Continuous
Delay	[0.0001, 0.01]	Continuous
Number of hidden layers	[1, 10]	Discrete with step = 1
Number of neurons in hidden layer	[1, 300]	Discrete with step = 1
Number of epochs	[5, 25]	Discrete with step = 5
Batch size	[100, 1,000]	Discrete with step = 100
Layer type	[1, 2]	Discrete with step = 1
Optimizer	[1, 6]	Discrete with step = 1
Initialization function	[1, 8]	Discrete with step = 1
Activation function	[1, 8]	Discrete with step = 1

Thus, the DNN hyperparameters and their domains are defined in this initialization stage. Twelve various DNN hyperparameters $N = 12$ are utilized. The hyperparameters of DNN and its respective domain along with the explanation are shown in **Table 4**.

Most of the hyperparameters utilized are numerical with an exception to layer type, activation function, and initialization for an optimizer as they are categorical. The indexing of all the feasible values to a successive number of ranges from one to that specific length of the list is done. Adagrad, Adam, Adamax, Nadam, RMS prop, and SGD are the elements included in the optimizer list. Dropout and Dense are the two elements present in the layer type. The elements of the initialization function include Normal, Uniform, He normal, He uniform, Zero, Glorot uniform, Glorot normal, and Leun uniform. Eight elements are present in the activation list such as linear, sigmoid, hard_sigmoid, Softmax, ReLU, Tanh, Soft plus, and Soft sign. The elements belonging to all the categorical hyperparameters are mentioned and explained in Keras execution. For every text, the optimization and result extraction stages will be implemented. The splitting of the data into two independent sets training and test sets T_i and Y_i are done by the optimization stage. From text to numerical values, the conversion of the training and test sets are performed and finally, they are normalized in [0,1]. Then to the PSO-based algorithm, these sets are given as input so that the optimized hyperparameters vector H_i are found. The construction of the DNN tuned H_i is done by the results extraction stage, then the DNN is trained on T_i and tested Y_i. The classification output values are extracted and processed. The execution of the finishing stage is done when all the text in the data is completed. The outcomes are obtained at the termination stage and at the end of the DNN experiment. The flow chart is explained in **Figure 4**.

RESULTS AND DISCUSSION

In this section, the evaluation criteria for text classification, the dataset explanation, and the results of the proposed deep learning architectures, followed by a comparison with other works are elaborated in detail.

Evaluation Criteria for Text Classification Results

The accuracy is mainly used as a classification criterion for the text classification results. In the fields of statistical classification, to evaluate its performance of it, this index is widely used.

The performance of the classifier is reflected by accuracy and is expressed as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{21}$$

The recall is expressed as:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{22}$$

The Precision is expressed as:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{23}$$

The geometric mean (g-mean) is expressed as follows:

$$gmean = \sqrt{\frac{TP \times TN}{(TP + FN) \times (TN + FP)}} \tag{24}$$

The Mathews Correlation Coefficient (MCC) is expressed as follows:

$$MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP + FN) \times (TP + FP) \times (TN + FP) \times (TN + FN)}} \tag{25}$$

Dataset Explanation

The methodology is tested on two datasets that are considered in our experiments, 20 newsgroups, and BBC news datasets. There are 20 different news comment groups in the 20 newsgroups dataset where each group indicates a news topic. Three different versions are present on the website (http://qwone.com/$\sim$jason/20Newsgroups/). The second version is selected where a total of 18,846 documents are found. The division of the dataset is done into two main parts, where for the train set, 11,314 documents are found and for the test set, 7,532 documents are found. The BBC news dataset consists of many news documents from the BBC website (http://www.bbc.co.uk/news/business/market_data/overview/). A total of 2,225 documents are included in the dataset which corresponds to five topics such as technology, business, politics, sports, and entertainment. For the train set, 1,600 documents are selected randomly and for the test set, 625 documents are selected.

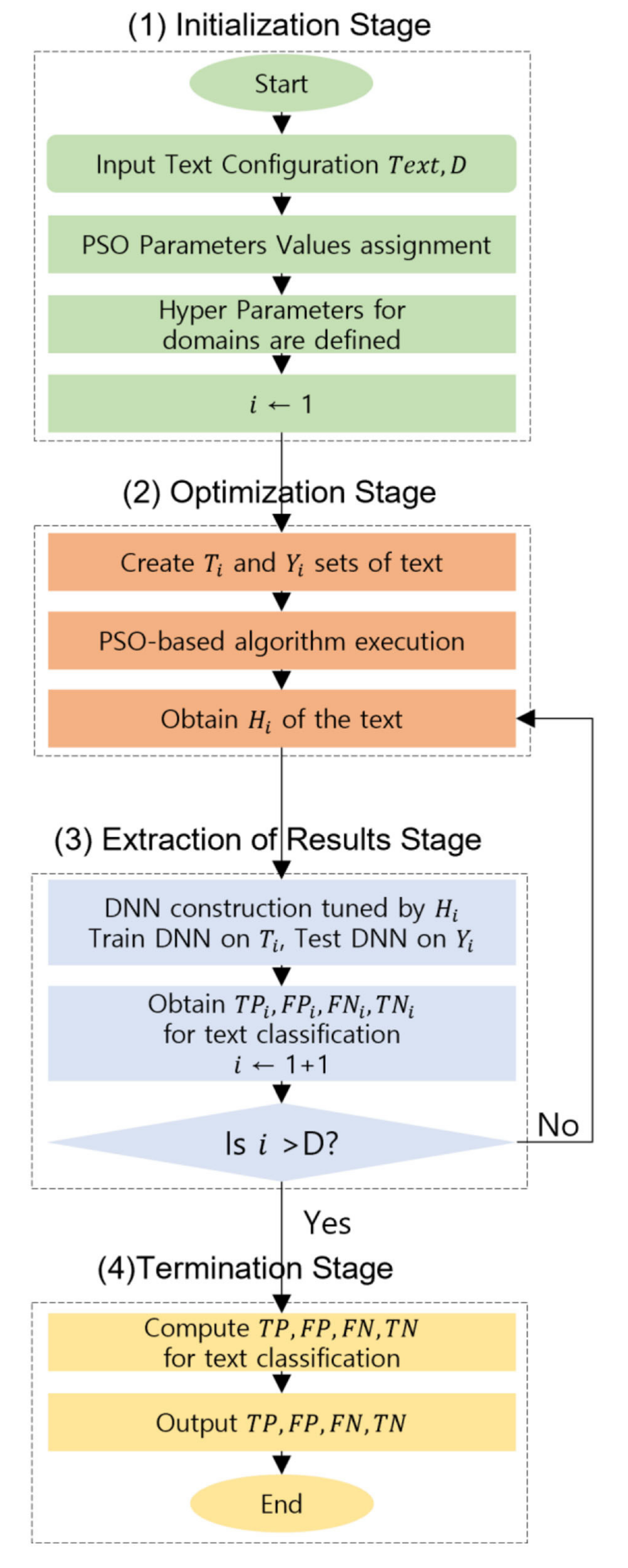
(1) Initialization Stage
Start
Input Text Configuration $Text, D$
PSO Parameters Values assignment
Hyper Parameters for domains are defined
$i \leftarrow 1$
(2) Optimization Stage
Create T_i and Y_i sets of text
PSO-based algorithm execution
Obtain H_i of the text
(3) Extraction of Results Stage
DNN construction tuned by H_i Train DNN on T_i, Test DNN on Y_i
Obtain TP_i, FP_i, FN_i, TN_i for text classification $i \leftarrow 1+1$
Is i >D?
No
Yes
(4)Termination Stage
Compute TP, FP, FN, TN for text classification
Output TP, FP, FN, TN
End

FIGURE 4 | Flow chart of the proposed Swarm DNN.

Results of the Proposed ECCNN and Swarm DNN Architectures

The effect of various parameters of the DE algorithm with the CCNN architecture is analyzed, such as population size, the main implementation procedure of DE, such as DE/best/1 or DE/best/2, mutation parameters F, crossover probability, and the results are shown in **Tables 5–10**. With the assignment of random values for more than a hundred possible combinations, finally, the six best results of the ECCNN combination are shortlisted and reported in **Tables 5–10**.

Table 5 explains the results of the ECCNN architecture with DE/best/1 and DE/best/2 strategy with population sizes of sixty and ninety along with F and CR set as 0.2 and 0.3, respectively. Under such a case, a classification accuracy of 79.22% for 20 newsgroup datasets for the DE/best/2 strategy and the second-highest classification accuracy of 78.91% with the DE/best/1 strategy was obtained again for 20 newsgroup datasets. For the BBC newsgroup dataset, a classification accuracy of 89.12% with DE/best/2 strategy and the second-highest classification accuracy of 87.63% with DE/best/1 strategy was obtained. **Table 6** explains the results of the CNN architecture with DE/best/1 and DE/best/2 strategy with population sizes of sixty and ninety along with F and CR set as 0.4 and 0.5, respectively. Under this case, the overall highest classification accuracy of 97.11% was obtained for the BBC newsgroup dataset with the DE/best/2 strategy, and overall high accuracy of 88.76% was obtained for the 20 newsgroup dataset with DE/best/2 strategy. **Table 7** explains the results of the ECCNN architecture with DE/best/1 and DE/best/2 strategy with population sizes of 60 and 90 along with F and CR set as 0.6 and 0.8, respectively. Under this case, a classification accuracy of 80.34% for the DE/best/2 strategy with a 20-newsgroup dataset was obtained and an accuracy of 91.37% with the DE/best/2 strategy with a BBC newsgroup dataset was obtained. **Table 8** explains the results of the ECCNN architecture with DE/best/1 and DE/best/2 strategy with population sizes of 60 and 90 along with F and CR set as 0.8 and 0.2, respectively. Under this case, a classification accuracy of 89.46% for DE/best/1 strategy with BBC newsgroup dataset was obtained and an accuracy of 84.76% for DE/best/2 strategy with 20 newsgroup dataset was obtained. **Table 9** explains the results of the ECCNN architecture with DE/best/1 and DE/best/2 strategy with population sizes of sixty and ninety along with F and CR set as 0.6 and 0.4, respectively. Under this case, a classification accuracy of 94.25% for DE/best/1 strategy with BBC newsgroup dataset was obtained and an accuracy of 82.36% for DE/best/1 strategy with 20 newsgroup dataset was obtained.

Table 10 explains the results of the ECCNN architecture with DE/best/1 and DE/best/2 strategy with population sizes of 60 and 90 along with F and CR set as 0.5 and 0.3, respectively. Under this case, a classification accuracy of 81.24% with DE/best/2 strategy with 20 newsgroup datasets was obtained and high accuracy of 91.48% for DE/best/1 strategy with BBC newsgroup dataset was obtained. **Table 11** shows the results of the Swarm DNN architecture for the parameters of the four different cases of values the highest classification accuracy of 97.32% was

TABLE 5 | Results of the ECCNN architecture with DE/best/1 and DE/best/2 strategy, F = 0.2, CR = 0.3.

DE parameters	DE/best/1	DE/best/1	DE/best/2	DE/best/2
	Population size 60	Population size 90	Population size 60	Population size 90
Dataset utilized	Accuracy	Accuracy	Accuracy	Accuracy
20 newsgroup	72.34	78.91	73.24	79.22
BBC newsgroup	87.63	82.35	81.56	89.12

TABLE 6 | Results of the ECCNN architecture with DE/best/1 and DE/best/2 strategy, F = 0.4, CR = 0.5.

DE Parameters	DE/best/1	DE/best/1	DE/best/2	DE/best/2
	Population size 60	Population size 90	Population size 60	Population size 90
Dataset utilized	Accuracy	Accuracy	Accuracy	Accuracy
20 newsgroup	81.62	84.45	85.56	88.76
BBC newsgroup	92.35	92.49	96.12	97.11

TABLE 7 | Results of the ECCNN architecture with DE/best/1 and DE/best/2 strategy, F = 0.6, CR = 0.8.

DE parameters	DE/best/1	DE/best/1	DE/best/2	DE/best/2
	Population size 60	Population size 90	Population size 60	Population size 90
Dataset utilized	Accuracy	Accuracy	Accuracy	Accuracy
20 newsgroup	79.12	77.45	80.34	78.35
BBC newsgroup	89.23	88.35	89.24	91.37

TABLE 8 | Results of the ECCNN architecture with DE/best/1 and DE/best/2 strategy, F = 0.8, CR =0.2.

DE parameters	DE/best/1	DE/best/1	DE/best/2	DE/best/2
	Population size 60	Population size 90	Population size 60	Population size 90
Dataset utilized	Accuracy	Accuracy	Accuracy	Accuracy
20 newsgroup	78.35	79.35	84.76	82.34
BBC newsgroup	88.45	89.46	85.56	85.13

TABLE 9 | Results of the ECCNN architecture with DE/best/1 and DE/best/2 strategy, F = 0.6, CR = 0.4.

DE parameters	DE/best/1	DE/best/1	DE/best/2	DE/best/2
	Population size 60	Population size 90	Population size 60	Population size 90
Dataset utilized	Accuracy	Accuracy	Accuracy	Accuracy
20 newsgroup	80.23	82.36	79.23	78.23
BBC newsgroup	91.04	94.25	89.94	92.46

TABLE 10 | Results of the ECCNN architecture with DE/best/1 and DE/best/2 strategy, F = 0.5, CR = 0.3.

DE parameters	DE/best/1	DE/best/1	DE/best/2	DE/best/2
	Population size 60	Population size 90	Population size 60	Population size 90
Dataset utilized	Accuracy	Accuracy	Accuracy	Accuracy
20 newsgroup	79.45	80.21	78.25	81.24
BBC newsgroup	91.48	89.24	84.59	87.31

obtained for Case 4 parameter values with the BBC news dataset and a high classification accuracy of 87.99% was obtained for Case 3 parameter values with 20 newsgroup datasets. The final comparison of the best results for the two developed deep learning models implemented for the two datasets is computed and tabulated in **Table 12**.

TABLE 11 | Results of the Swarm DNN architecture for the parameters of the four different cases.

PSO parameters	PSO Parameters: Case 1	PSO Parameters: Case 2	PSO Parameters: Case 3	PSO Parameters: Case 4
Dataset utilized	Accuracy	Accuracy	Accuracy	Accuracy
20 newsgroup	84.45	82.21	87.99	85.24
BBC newsgroup	93.48	93.24	95.59	97.32

TABLE 12 | Consolidated result analysis of the proposed techniques in text classification.

Dataset	Model	Accuracy	Precision	Recall	Geometric mean	MCC
20 newsgroup	ECCNN	88.76	82.45	81.35	81.23	79.45
	Swarm DNN	87.99	83.13	82.45	82.06	80.13
BBC	ECCNN	97.11	91.36	90.43	89.13	88.31
newsgroup	Swarm DNN	97.32	90.18	89.48	88.12	87.06

From **Table 12**, it is understood that when the proposed ECCNN model is implemented with the DE/best/2 strategy and with an F and CR values of 0.4 and 0.5 respectively, a classification accuracy of 88.76%, precision of 82.45%, Recall of 81.35%, the geometric mean of 81.23% along with an MCC of 79.45% is obtained for the 20 Newsgroup dataset. Similarly, when the proposed ECCNN model is implemented with the DE/best/2 strategy and with an F and CR values of 0.4 and 0.5 respectively, a classification accuracy of 97.11%, precision of 91.36%, Recall of 90.43%, Geometric mean of 89.13% along with an MCC of 88.31% is obtained for the BBC Newsgroup dataset.

Similarly, when the proposed Swarm DNN model is implemented with PSO parameters pertaining to case 4, a classification accuracy of 87.99%, precision of 83.13%, Recall of 82.45%, and Geometric mean of 82.06% along with an MCC of 80.13% is obtained for the 20 Newsgroup dataset. Similarly, when the proposed Swarm DNN is implemented with PSO parameters pertaining to case 4, a classification accuracy of 97.32%, precision of 90.18%, Recall of 89.48%, and Geometric mean of 88.12% along with an MCC of 87.06% is obtained for the BBC Newsgroup dataset. As far as statistical tests are concerned, initially, a 2-sided Wilcoxon rank-sum test was conducted and the ρ value obtained was < 0.05, thereby obtaining a high confidence level. A Kruskal Wallis test too was analyzed and the value ρ obtained was <0.01 proving its statistical significance and correctness. A Friedman test too was conducted to analyze the unique differences between multiple algorithms of the datasets and upon testing there were clear and distinct variations. Finally, a Cohen's Kappa coefficient test too was conducted, and the values always reached a good agreement or a very good agreement category.

Comparison With Other Works Utilizing the Same Database

Some of the commonly used databases for text classification include Standard Reuters 21578, MNIST dataset, MIT newspaper, Wikipedia XML Corpus, MTI ML Site, Jeeves Support System database, GENIA and CRAFT, Nature Database, Dream bank Report Corpus, along with the databases used in this paper such as 20 newsgroups and BBC newsdata (Altinel and Ganiz, 2018). Very few peer-reviewed works have been reported in high-quality journals and literature with the 20 newsgroups and BBC newsdata and some important works are considered here for comparison with the results obtained in this work. A vigorous text classifier dependent on a Denoising Deep Neural Network (DDNN) was proposed by Aziguli and they reported a classification accuracy of 92.86% for the BBC newsgroup dataset and 73.78% for the 20-newsgroup dataset (Aziguli et al., 2017). A multilayer classification was utilized by Pradhan et al. reporting a classification accuracy of 97.67% for BBC news and 86.70% for 20 newsgroup datasets (Pradhan et al., 2017). An Instance-Infused LSTM was proposed by Chowdhury et al. reporting an accuracy of 78.29% for the News 20 dataset and 96.09% for the BBC news dataset (Chowdhury et al., 2019). A combination of CNN with Long Short-Term Memory (LSTM) was utilized by Camacho et al. reporting 97% accuracy for the BBC news group set and 90.7% for the 20 newsgroup sets (Camacho-Collados and Pilehvar, 2017). A Deep Belief Network (DBN) with Softmax model (Jiang et al., 2018), deep learning with meta-thesaurus (Liu et al., 2017), and LSTM (Shih et al., 2017) were utilized giving a classification accuracy of 85.57%, 69.82%, and 86.2% for the 20-newsgroup dataset, respectively. Support Vector Machine (SVM) and Naïve Bayesian Classifier (NBC) were utilized by Shirsat reporting a classification accuracy of 96.46% with SVM and 94.16% with NBC for the BBC news dataset (Shirsat et al., 2019). A bigram alphabet approach for text representation and classification of the BBC news dataset was reported by Elghannam reporting a classification accuracy of 92.6% (Elghannam, 2019). However, in our work, we introduced swarm intelligence to the modified deep learning models and developed interesting results. By and large, the work produced a higher classification accuracy of 97.32% using the Swarm DNN model and 97.11% using the ECCNN model for BBC datasets, respectively. Similarly, this work reports a high classification accuracy of 87.99% using the Swarm DNN model and 88.76% using the ECCNN model for 20-news group datasets respectively, which is quite a commendable performance when compared to other works. As the methods proposed in this work are quite interesting and easy to implement, these models can be very well applied to other datasets too.

CONCLUSION AND FUTURE WORK

In the area of NLP, text classification has been a very interesting issue. For big data analysis, good classification accuracy is very important to implement NLP for scientific data analytics. To manage a huge amount of text documents in the fields of web mining, information retrieval, natural language processing, and content security research areas, text classification plays a huge role. The assignment of one or more predefined classes to a natural text document based on the knowledge obtained from text expression is the vital task of text classification. Therefore, the development of effective and versatile algorithms is quite challenging in the area of text classification because of the large size of the text data. In this work, two successful deep learning models such as ECCNN and Swarm DNN were developed and tested on two datasets and the results were obtained. A very high classification accuracy of 97.32% was obtained for the BBC newsgroup dataset when utilized with the Swarm DNN model. The second-highest classification accuracy of 97.11% was obtained for the BBC newsgroup dataset with the ECCNN model, where the DE strategy was DE/best/2 and with a specific set of assigned mutation scaling and crossover probability parameters. The third-highest classification accuracy rate of 87.99% was obtained for the 20-newsgroup dataset when the Swarm DNN model was utilized. The fourth-highest classification accuracy of 88.76% was obtained for the 20-newsgroup dataset when the ECCNN model was utilized. Future works aim to work with many other nature-inspired and ensemble deep learning models for efficient text classification purposes.

DATA AVAILABILITY STATEMENT

All the relevant programming codes implemented in the work can be obtained from the corresponding author upon request.

AUTHOR CONTRIBUTIONS

All authors listed have made a substantial, direct, and intellectual contribution to the work and approved it for publication.

FUNDING

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2021-0-02068, Artificial Intelligence Innovation Hub) and partly supported and funded by the Korean National Police Agency (Project Name: AI-based Crime Investigation Support System / Project Number: PR10-02-000-21).

REFERENCES

Agarwal, B., Mittal, N., Bansal, P., and Garg, S. (2015). Sentiment analysis using common-sense and context information. *Comput. Intell. Neurosci*. 2015, 9. doi: 10.1155/2015/715730

Aggarwal, C., and Zhai, C. (2012). "A survey of text classification algorithms," in *Mining Text Data*. Springer: Boston, MA, USA. p. 163–222. doi: 10.1007/978-1-4614-3223-4_6

Alhudhaif, A. S. (2022). A particle swarm optimization based deep learning model for vehicle classification. *Comput. Syst. Sci. Eng.* 40, 223–235. doi: 10.32604/csse.2022.018430

Altinel, M., and Ganiz, C. (2018). Semantic text classification: a survey of past and recent advances. *Inf Process Manag*. 54, 1129–1153, doi: 10.1016/j.ipm.2018.08.001

Amini, M., Usunier, N., and Goutte, C. (2009). "Learning from multiple partially observed views-an application to multilingual text categorization," in *Proceedings of the Advances in Neural Information Processing Systems.* Vancouver, Canada. p. 28–36.

Aro, T. O., Akande, H. B., Adewole, K. S., Aregbesola, K. M., and Jibrin, M. B. (2020). Ehanced textual data classification using Particle Swarm Optimization (PSO) algorithm. *J. ICT Res. Appl.* 2, 1–14.

Asif, M., Nagra, A. A., and Ahmad, M. B. (2022). Feature selection empowered by self-inertia weight adaptive particle swarm optimization for text classification. *Appl. Artif. Intell*. 36, 1. doi: 10.1080/08839514.2021.2004345

Aziguli, W., Zhang, Y., Xie, Y., Zhang, D., Luo, X., Li, C., et al. (2017). A robust text classifier based on denoising deep neural network in the analysis of big data. *Scientific Programming*. 2017. doi: 10.1155/2017/3610378

Baioletti, M., Bari, G., Milani, A., and Poggioni, V. (2020). Differential evolution for neural networks optimization. *Mathematics*. 8, 69. doi: 10.3390/math8010069

Camacho-Collados, J., and Pilehvar, M. T. (2017). On the role of text preprocessing in neural network architectures: An evaluation study on text categorization and sentiment analysis. *arXiv*. 1707, 01780. doi: 10.18653/v1/W18-5406

Chen, C., and Dai, J. (2021). Mitigating backdoor attacks in lstm-based text classification systems by backdoor keyword identification. *Neurocomputing*. 452, 253–262. doi: 10.1016/j.neucom.2021.04.105

Chen, K., Zhang, Z., Long, J., and Zhang, H. (2016). Turning from TF-IDF to TF-IGM for term weighting in text classification. *Expert Syst. Appl.* 66, 245–260. doi: 10.1016/j.eswa.2016.09.009

Chen, Y. W., Zhou, Q., Luo, W., and Du, J. X. (2016). Classification of Chinese texts based on recognition of semantic topics. *Cognit. Computat*. 8, 114–124. doi: 10.1007/s12559-015-9346-8

Chowdhury, S. B. R., Annervaz, K. M., and Dukkipati, A. (2019). "Instance based inductive deep transfer learning by cross-dataset querying with locality sensitive hashing," in *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resources NLP*. Hong Kong, China. p. 183–191.

Comite, F. D., Gilleron, R., and Tommasi, M. (2003). "Learning multi- ' label alternating decision trees from texts and data," in *Machine Learning and Data Mining in Pattern Recognition*. Springer: Berlin, Germany. p. 35–49. doi: 10.1007/3-540-45065-3_4

Conneau, H., Barrault, S., and Lecun, Y. (2017). *Very Deep Convolutional Networks for Text Classification.* Available online at: http://arxiv.org/abs/1606, 01781v.2. doi: 10.18653/v1/E17-1104

Deng, J., Cheng, L., and Wang, Z. (2021). Attention-based bilstm fused cnn with gating mechanism model for Chinese long text classification. *Comput. Speech. Lang*. 68, 101182. doi: 10.1016/j.csl.2020.101182

Diab, M., and El Hindi, K. M. (2017). "Using differential evolution for fine tuning naïve Bayesian classifiers and its application for text classification," *Appl. Soft Comput*. 54, 183–199. doi: 10.1016/j.asoc.2016.12.043

Dixit, A., and Bansal, M. (2020). Feature selection for text and image data using differential evolution with svm and naïve bayes classifiers'. *Eng. J.* 24, 2020 doi: 10.4186/ej.2020.24.5.161

Elghannam (2019). Text representation and classification based on bi-gram alphabet. *J. King Saud Univ. - Comput. Inf. Sci*. 2019.

Gracia-mendoza, C., Gambino, O. J., Villarreal-Cervantes, M., Calvo, H. (2020). Evolutionary optimization of ensemble learning to determine sentiment polarity in an unbalanced multiclass corpus. *Entropy (Basel, Switzerland).* 22, 12. doi: 10.3390/e22091020

Guo, L., Zhang, D., Wang, L., Wang, H., and Cui, B. (2018). CRAN: a hybrid CNN-RNN attention-based model for text classification. *Conceptual Model.* 5, 571–585. doi: 10.1007/978-3-030-00847-5_42

Han, E. H. S., Karypis, G., and Kumar, V. (2001). *Text Categorization Using Weight Adjusted K-Nearest Neighbor Classification*. Springer: Berlin, Germany. doi: 10.1007/3-540-45357-1_9

Han, H., Liu, G., and Dang, J. (2019). An interactive model of target and context for aspect-level sentiment classification. *Comput. Intell. Neurosci*. 2019, 8. doi: 10.1155/2019/3831809

Ifrim, G., Bakir, G., and Weikum, G. (2008). F "ast logistic regression for text categorization with variable-length n-grams, KDD '08," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. p. 354–362. doi: 10.1145/1401890.1401936

Jeong, J.-., H., Shim, K-, H., Kim, D.-., J., and Lee, S-, W. (2020). Brain-controlled robotic arm system based on multi-directional CNN-BiLSTM network using EEG signals. *IEEE Trans. Neural Syst*. 28, 1226–1238. doi: 10.1109/TNSRE.2020.2981659

Jiang, M., Liang, Y., Feng, X., Fan, X., Pei, Z., Xue, Y., et al. (2018). Text classification based on deep belief network and softmax regression. Neural. Comput. Appl. 29, 61–70. doi: 10.1007/s00521-016-2401-x

Johnson, R., and Zhang, T. (2015). "Effective use of word order for text categorization with convolutional neural networks," *in Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Denver, CO, USA. doi: 10.3115/v1/N15-1011

Joulin, E., Bojanowski, G., and Mikolov, T. (2017). "Bag of tricks for efficient text classification," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*. EACL. 427–431. doi: 10.18653/v1/E17-2068

Junior, F. E. F., and Yen, G. G. (2019). Particle swarm optimization of deep neural networks architectures for image classification. *Swarm Evol. Comput*. 49, 62–74 doi: 10.1016/j.swevo.2019.05.010

Kaliyapillai, S., and Krishnamurthy, S. (2020). Differential evolution based hyperparameters tuned deep learning models for disease diagnosis and classification. *Adv. Sci. Technol. Eng. Syst. J*. 5, 253–261. doi: 10.25046/aj050531

Kennedy, J., and Eberhart, R. (1995). Particle swarm optimization. *Proc. Int. Jt. Conf. Neural Netw*. 4, 1942–1948.

Kilimci, Z. H., and Akyokus, S. (2018). Deep learning-and word embedding-based heterogeneous classifier ensembles for text classification. *Complexity*. 2, 14–18. doi: 10.1155/2018/7130146

Kowsari, K., Brown, D. E., Heidarysafa, M., Jafari Meimandi, K., Gerber, M. S., and Barnes, L. E. (2017). "HDLTex: Hierarchical Deep Learning for Text Classification," in *16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. Cancun, Mexico. p. 364–371.

Lai, S., Xu, L., Liu, K., and Zhao, J. (2015). Recurrent convolutional neural networks for text classification. *AAAI*. 333, 2267–2273.

Lewis, D. (1992). "Evaluation of phrasal and clustered representations on a text categorization task," in *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. p. 37–50. doi: 10.1145/133160.133172

Li, L., Yu, S., Zhong, L., and Li, X. (2015). Multilingual text detection with nonlinear neural network. *Math. Probl. Eng*.2015, 7. doi: 10.1155/2015/431608

Li, Q., Peng, H., and Xi, J. (2020). A survey on text classification: from shallow to deep learning. *IEEE Trans Neural Netw Learn Syst*. 31, 11.

Liu, M., Haffari, G., Buntine, W., and Ananda-Rajah, M. (2017). Leveraging linguistic resources for improving neural text classification. *Proc. Australasian Lang. Technol. Associat. Workshop*. 34–42.

Liu, M., Liu, L., Cao, J., and Du, Q. (2021). Co-attention network with label embedding for text classification. *Neurocomputing*. 471, 61–69. doi: 10.1016/j.neucom.2021.03.078

Liu, P., Qiu, X., and Huang, X. (2016). *Recurrent Neural Network for Text Classification with Multi-Task Learning*. Available online at: https://arxiv.org/abs/1605.05101v1.

Liu, R., Liu, Y., Yan, Y., and Wang, J. Y. (2020). Iterative deep neighborhood: a deep learning model which involves both input data points and their neighbors. *Comput. Intell. Neurosci*. 2020, 10. doi: 10.1155/2020/9868017

Liu, X., Zhang, C., Cai, Z., Yang, J., Zhou, Z., and Gong, X. (2021). Continuous particle swarm optimization-based deep learning architecture search for hyperspectral image classification. *Remote Sens*. 13, 1082. doi: 10.3390/rs13061082

Lorenzo, P. R., Nalepa, J., Kawulok, M., Ramos, L. S., and Pastor, J. R. (2017a). "Particle Swarm Optimization for hyper-parameter selection in deep neural networks," in *GECCO'17: Proceedings of the Genetic and Evolutionary Computation Conference*.p. 481–488. doi: 10.1145/3071178.3071208

Lorenzo, P. R., Nalepa, J., Ramos, L. S., and Pastor, J. R. (2017b). "Hyper-parameter selection in deep neural networks using parallel particle swarm optimization," in *GECCO'17: Proceedings of the Genetic and Evolutionary Computation Conference*. p. 1864–1871. doi: 10.1145/3067695.3084211

Ma, Y., Liu, X., Zhao, L., Liang, Y., Zhang, P., and Jin, B. (2022). Hybrid embedding-based text representation for hierarchical multi-label text classification. *Expert Syst. Appl*. 187, 115905, (2022). doi: 10.1016/j.eswa.2021.115905

McCallum, and Nigam, K. A. (1998). "Comparison of event models for naive Bayes text classification," in *AAAI-98 Workshop on Learning for Text Categorization*. Wisconsin, USA: AAAI Press. p. 41–48.

Mohamed, W. (2015). An improved differential evolution algorithm with triangular mutation for global numerical optimization. *Comput Ind. Eng*. 85, 359–375. doi: 10.1016/j.cie.2015.04.012

Murthy, S. K. (1998). Automatic construction of decision trees from data: a multi-disciplinary survey. *Data Min Knowl Discov*. 2, 345–389. doi: 10.1023/A:1009744630224

Onan, S. Korukoglu, and Bulut, H. (2016). ".Ensemble of keyword extraction methods and classifiers in text classification. *Expert Syst. Appl*. 57, 232–247. doi: 10.1016/j.eswa.2016.03.045

Post, M., and Bergsma, S. (2013). "Explicit and implicit syntactic features for text classification," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. p. 866–872.

Pradhan, L., Taneja, N. A., Dixit, C., and Suhag, M. (2017). Comparison of text classifiers on news articles. *Int. Res. J. Eng. Technol*. 4, 2513–2517.

Ren, J., Wu, W., Liu, G., Chen, Z., and Wang, R. (2021). Bidirectional gated temporal convolution with attention for text classification. *Neurocomputing*. 455, 265–273. doi: 10.1016/j.neucom.2021.05.072

Santos, N. D., and Gatti, M. (2014). Deep convolutional neural networks for sentiment analysis of short texts," in *the 25th International Conference on Computational Linguistics*. Dublin, Ireland. p. 69–78.

Schmidhuber, J. (2015). Deep learning in neural networks: an overview. *Neural Networks*. 61, 85–117. doi: 10.1016/j.neunet.2014.09.003

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Comput. Surv*. doi: 10.1145/505282.505283

Shih, C. H., Yan, B. C., Liu, S. H., and Chen, B. (2017). Investigating Siamese LSTM networks for text categorization, in 2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC). *IEEE*. 641–646. doi: 10.1109/APSIPA.2017.8282104

Shirsat, V. S., Jagdale, R. S., and Deshmukh, S. N. (2019). "Sentence level sentiment identification and calculation from news articles using machine learning techniques," in *Computing, Communication and Signal Processing*. Springer. p. 371–376. doi: 10.1007/978-981-13-1513-8_39

Sikdar, U. K., Ekbal, A., and Saha, S. (2012). "Differential Evolution based Feature Selection and Classifier Ensemble for Named Entity Recognition," in *Proceedings of COLING 2012: Technical Papers*. p. 2475–2490.

Sun, E., Lim, P., and Liu, Y. (2009). On strategies for imbalanced text classification using SVM: a comparative study. *Decis. Support Syst*. 48, 191–201. doi: 10.1016/j.dss.2009.07.011

Sun, N., and Du, C. (2021). News Text classification method and simulation based on the hybrid deep learning model. *Complexity*. 2021, 8064579 doi: 10.1155/2021/8064579

Thirumoorthy, K., and Muneeswaran, K. (2021). Feature selection using hybrid poor and rich optimization algorithm for text classification. *Patt. Recognit. Lett*. 147, 63–70. doi: 10.1016/j.patrec.2021.03.034

Tuyen, D. N., Tuan, T. M., Son, L. H., Ngan, T., Giang, N. L., Thong, P. H., et al. (2021). A novel approach combining particle swarm optimization and deep learning for flash flood detection from satellite images. *Mathematics*. 9, 2846. doi: 10.3390/math9222846

Wang, X., Luo, B., Zhang, Q. B., and Wei, S. (2004). Estimation for the number of components in a mixture model using stepwise split-and-merge EM algorithm. *Pattern Recognit. Lett*. 25, 1799–1809. doi: 10.1016/j.patrec.2004.07.007

Yan, and Guo, S. (2019). Leveraging contextual sentences for text classification by using a neural attention model. *Comput. Intell. Neurosci*. 2019, 11. doi: 10.1155/2019/8320316

Yao, L., Mao, C., and Luo, Y. (2019). Graph convolutional networks for text classification. *The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)*. 7370–8. doi: 10.1609/aaai.v33i01.33017370

Zahran, B. M., and Kanaan, G. (2009). Text feature selection using particle swarm optimization algorithm. *World Appl Sci J.* 2009, 69–74.

Zhang, X., Xu, J., Soh, C., and Chen, L. (2022). La-hcn: label-based attention for hierarchical multi-label text classification neural network. *Expert Syst. Appl.* 187, 115922. doi: 10.1016/j.eswa.2021.115922

Zhang, X., and Zhao, J., and, Y., LeCun (2015). "Character-level convolutional networks for text classification," in *Advances in Neural Information Processing Systems*. p. 649–657.

Zheng, J., Guo, Y., Feng, C., and Chen, H. (2018). A hierarchical neural-network-based document representation approach for text classification. *Math. Probl. Eng.* 2018, 10. doi: 10.1155/2018/7987691

Zheng, S., Hongyun Bao, J. X., Hao, Y., Qi, Z., and Hao, H. (2016). "A bidirectional hierarchical skip-gram model for text topic embedding," in *2016 International Joint Conference on Neural Networks (IJCNN),* Vancouver, BC. p. 855–862.

Zhou, C. S. (2015). Neural network for text classification. *arXiv arXiv*. 1511, 08630.

Zhou, P., Qi, Z., Zheng, S., Xu, J., Bao, H., and Xu, B. (2016). "Text classification improved by integrating bidirectional lstm with two-dimensional max pooling," *26th International Conference on Computational Linguistics (COLING).* Osaka, Japan. p. 3485–3495.

Zhu, Y., Zheng, W., and Tang, H. (2020). Interactive dual attention network for text sentiment classification. *Comput. Intell. Neurosci*. 2020, 11. doi: 10.1155/2020/8858717

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.