Received April 2, 2021, accepted May 22, 2021, date of publication May 27, 2021, date of current version June 7, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3084131

Immunocomputing-Based Approach for Optimizing the Topologies of LSTM Networks

ALI AL BATAINEH, (Member, IEEE), AND DEVINDER KAUR, (Life Senior Member, IEEE)
Electrical Engineering and Computer Science Department, The University of Toledo, Toledo, OH 43606, USA
Corresponding author: Ali Al Bataineh (ali.albataineh@utoledo.edu)

ABSTRACT This paper aims to automatically design optimal LSTM topologies using the clonal selection algorithm (CSA) to solve text classification tasks such as sentiment analysis and SMS spam classification. Designing optimal topologies involves determining the best configuration of hyperparameters that will give the best performance. The current state-of-the-art LSTM topologies are often designed using trial and error approaches which are incredibly time-consuming and require domain experts. Our proposed method, referred to as CSA-LSTM, is evaluated using the Large Movie Review Dataset (IMDB). Furthermore, to verify the robustness of the hyperparameters discovered by CSA for the IMDB dataset, we have used them for the other datasets, viz. the Twitter US Airline Sentiment and the SMS Spam Collection. Additionally, the discovered hyperparameters for the LSTM are combined with pre-determined convolutional neural network (CNN) layers to achieve the same or better results to fast the training time and fewer trainable parameters. For further verification and evaluation of the generalization ability and effectiveness of the proposed approach, it is compared with four machine learning algorithms widely used for text classification tasks: (1) random forest (RF), (2) logistic regression (LR), (3) support vector machine (SVM), and (4) multinomial naive Bayes (NB). The results of our experiments show that the LSTM topologies automatically designed by our CSA method are less expensive, reusable and outperform the machine learning algorithms and other models in the literature evaluated on the same three datasets. Through our proposed method, LSTM's best topology can be self-determined without any human intervention, making CSA-based algorithms a promising approach to automatically design optimal LSTM topologies that provide the best performance for a given task.

INDEX TERMS Clonal selection algorithm, deep learning, hyperparameters optimization, IMDB, immunocomputing, long short-term memory.

I. INTRODUCTION

Today, deep learning is one of the most up-and-coming technologies that are mainly driving the modern rise of artificial intelligence (AI) and machine learning [1], [2]. With the significant advances in technology and algorithms in the last few years, deep learning has paved the way for a new generation of AI applications [3]. In many of these applications, i.e., classification from text, sound, or images, the performance of deep learning algorithms matched and sometimes exceeding human-level performance [4], [5]. Deep learning refers to neural networks with several hidden layers [6], [7]. Deep neural networks perform computing tasks similar to biological neurons in the human brain [8]. One of the most commonly used algorithms behind the scenes of the amazing successes seen in deep learning over the past few years is long short-term memory networks, a.k.a LSTMs [9]. These networks provide a lot of flexibility and have proven themselves to be the state-of-the-art solution to a wide range of sequence prediction problems such as natural language processing (NLP) [10], machine translation [11], sentiment analysis [12], image captioning [13], and speech recognition [14]. LSTM networks are powerful and robust variants of recurrent neural networks (RNN) and belong to the most promising algorithms because they are the only network with internal memory, enabling them to remember long sequence patterns. Recently, there has been a lot of demand to design optimal LSTM topologies automatically. The right network topology has a big impact on the performance of deep learning models. Unlike traditional machine learning models, deep learning models contain many hyperparameters, and the designer must configure their values before training [15]. Some of which are significantly essential hyperparameters whose setting can have a substantial impact on network performance. As a

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

matter of fact, there may be an LSTM topology that provides very accurate results for a given problem; however, if the number of LSTM units is slightly increased, then the entire topology may perform poorly on the same problem.

The optimal topologies are usually designed manually by experts or by means of heuristics. The grid search approach [16] is common to use when the number of dimensions (or hyperparameters) is small. Although it guarantees to find the best configuration, it is still not preferable for search space that contains many hyperparameters. A better alternative in terms of time complexity to use is the random search approach in which a combination of the hyperparameters is picked randomly from the configuration space. However, the random search approach is not guaranteed to find the best hyperparameters as it does not explore all the possible combinations. Although both grid search and random search approaches perform better than the manual tuning approach in most cases, one common drawback of both methods is that they do not consider the previously chosen hyperparameters' performance while choosing the next hyperparameters set. Bayesian optimization is another popular approach that has been used to direct a search for deep learning model topologies. Unlike the grid search and random search approaches, the Bayesian approach considers past evaluation results while choosing a new hyperparameter set. But it is still hard to implement the Bayesian approach in practice [17].

Because of these challenges, there have been ever-growing demand to automate the process of designing deep learning topologies in a more efficient way in terms of the solution obtained and the time complexity. We propose a simple yet effective method based on the clonal selection algorithms (CSA) to describe the above problems. CSA is an effective technique fairly easy to search for optimal LSTM topologies automatically. CSA has shown to deliver a more robust and effective approach to solving optimization problems [18], [19]. It is capable of finding a global solution by employing biologically inspired operators such as selection, cloning, and hypermutation [20], [21]. This motivates us to employ it to automate the challenging process of designing LSTM topologies.

In a nutshell, the following contributions have been delivered:

1) This work is the first to use a CSA-based method to automatically search for optimal, reusable LSTM topology for solving text classification tasks such as sentiment analysis and SMS spam classification.
2) This work has not used any regularization method, which proves the effectiveness of CSA to discover optimal topologies that can avoid overfitting.
3) We propose a simple integer encoding scheme to encode the all the possible hyperparameters and the layers of LSTM topology, allowing efficient implementation of different types of mutation like the truncated Gaussian mutation.
4) Our proposed method outperforms the proposed machine learning algorithms and other models in the literature on the three real-world datasets.

This paper is organized as follows. Section II presents related work, describes the current limitations and highlights our method's contributions to the area. Section III provides a background of LSTMs and CSA, respectively. Section IV discusses in great detail the implementation of the proposed CSA-based method to design LSTM topologies automatically. Experiment design and results are provided in Sections V and VI, respectively. Finally, conclusive remarks for the current work and potential future ambitions in which this research could advance are given in Section VII.

II. RELATED WORK

A few recent works have investigated the use of nature-inspired algorithms to automatically search for optimal LSTM based RNN topologies. In [22], the authors proposed a hybrid approach that integrates the LSTM network and genetic algorithm (GA) for stock market forecasting. GA was used to search for the optimal value for the time window size and number of LSTM units. They concluded that the GA-LSTM approach is an effective stock market forecasting method to reflect temporal patterns and outperform standard models.

In [23], the authors adopted the GA method to optimize LSTM topologies for water temperature prediction in urban rivers. Like [22], the GA was only used to find the optimal values of the time window size and the number of LSTM units. Their findings reported that the hybrid model of the GA-LSTM network outperformed traditional RNNs and thus can be used as an advanced deep learning system for time series analysis.

In [24], the authors proposed an approach based on evolutionary optimization techniques (i.e., GA and genetic programming) for designing LSTM topologies. Experiments were performed with three public word-processing datasets for part-of-speech tagging. The findings showed that the proposed approach is robust and outperformed random search methods.

In [25], the authors proposed the particle swarm optimization (PSO) algorithm to design the LSTM topology, which involves finding the optimal values for only three hyperparameters: number of hidden neurons, activation function and learning rate. Experiments were conducted with educational datasets. However, no experiments compare their proposed method to other neural topology search methods. So, it not possible to draw any conclusions regarding the actual benefits acquired from their proposed method.

In [26], the authors proposed a framework to automatically search for optimal LSTM hyperparameters (batch size and the number of hidden neurons) using differential evolution (DE) for emotion classification. They evaluated and compared their proposed framework with other hyperparameter search methods such as PSO, simulated annealing, and random search using a dataset they collected from wearable sensors. Experimental results showed that DE is competitive in finding the optimal LSTM hyperparameter values but is computationally expensive.

In [27], the authors explored using the ant colony optimization (ACO) algorithm to optimize the best previously used LSTM RNN fixed topology to predict excessive turbine engine vibration events. The evolved LSTM topology using ACO reduced the engine vibration mean prediction error and the number of trainable weights of the LSTM network.

In [28], the authors proposed the artificial bee colony (ABC) algorithm, which imitates the behavior of bee colonies in foraging to search for the optimal values of the hyperparameters of the LSTM topology for bitcoin price prediction. The finding of this study showed that the LSTM model evolved using the ABC outperformed handcrafted LSTM models.

As can be seen from the related work, most research has focused on designing LSTM topologies for only one specific dataset and one type of problem, i.e., time series forecasting. Our work expands this and discovers LSTM topologies using CSA to work well for multiple challenging text classification datasets. Additionally, in most cases, the related work took into account only two hyperparameters (e.g., window size and the number of LSTM units) for optimization. Besides the number of LSTM units, our work includes more hyperparameters to optimize, for instance, the number of epochs, batch size, optimizer, possibility to add fully connected layers and the number of their neurons. Finally, our work proposes adding CNN layers to reduce the number of trainable parameters (weights and biases), which can reduce the training time and, at the same time, increase the model prediction accuracy.

III. BACKGROUND

A. LSTM

LSTM networks were designed by Sepp Hochreiter and Juergen Schmidhuber to overcome the vanishing gradients problem that can be encountered when training conventional RNN models. In their paper [29], they work to address the problem of long-term dependencies. LSTM networks also have the chain-like topology of regular RNNs, but LSTM units are used as the internal building units for the layers of RNNs to extend the memory to remember inputs over a long period of time. Each LSTM cell has the same inputs and outputs as a regular RNN but has more parameters and a gating system to regulate the information flow. The most important component in LSTM is the cell state which carries relevant information via functional units called gates. There are three gates: a forget gate, an input gate, and an output gate. These gates decide which information to keep or throw away during the training process based on their importance to make predictions. Figure 1 shows an illustration of an LSTM network followed by an explanation for each of the three gates [30].

1) FORGET GATE

This gate decides what information should be forgotten (thrown away) or kept. The decision is made by passing

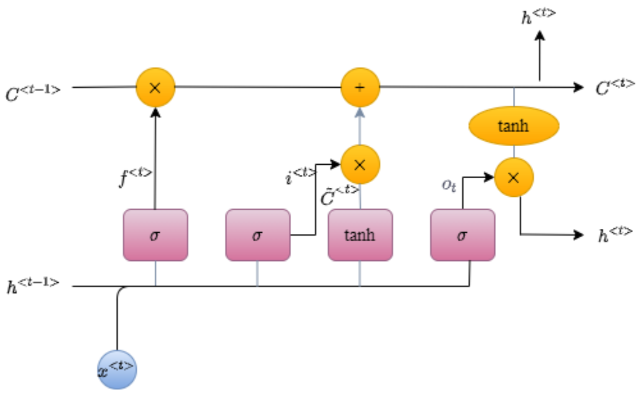

FIGURE 1. LSTM network.

the previous hidden state $h^{<t-1>}$ and the current input $x^{<t>}$ through the sigmoid function (σ). The output of this gate is given in E.q (1); the closer to 1 means to keep, and the closer to 0 means to forget.

$$f^{<t>} = \sigma\left(W^{<f>} \cdot \left[h^{<t-1>}, x^{<t>}\right] + b^{<t>}\right) \tag{1}$$

where $W^{<f>}$ and $b^{<t>}$ are parameters specific to the forget gate.

2) INPUT GATE

This gate is responsible for adding information to the cell state. The addition of information is a three-step process:

1) A sigmoid function is used to regulate what values need to be added to the cell state.

$$i^{<t>} = \sigma\left(W^{<i>} \cdot \left[h^{<t-1>}, x^{<t>}\right] + b^{<i>}\right) \tag{2}$$

where $W^{<i>}$ and $b^{<i>}$ are parameters specific to the input gate.

2) Creating a vector of new candidate values that could be added to the cell state using the tanh function.

$$\tilde{C}^{<t>} = \tanh\left(W^{<c>} \cdot \left[h^{<t-1>}, x^{<t>}\right] + b^{<c>}\right) \tag{3}$$

3) Multiplying the old cell state ($C^{<t-1>}$) with the forget gate $f^{<t>}$ and then we add $i^{<t>} * \tilde{C}^{<t>}$ to update the cell state ($C^{<t>}$).

$$C^{<t>} = f^{<t>} * C^{<t-1>} + i^{<t>} * \tilde{C}^{<t>} \tag{4}$$

This three-step process ensures that only important and not redundant information is added to the cell state.

3) OUTPUT GATE

This gate selects and outputs necessary information. The functioning of the output gate can be divided into two steps:

1) Employ a sigmoid function to decide what parts of the cell state to output.

$$o^{<t>} = \sigma\left(W^{<o>} \left[h^{<t-1>}, x^{<t>}\right] + b^{<o>}\right) \tag{5}$$

2) Pass the cell state through a tan function and multiply it with $o^{<t>}$ to decide what parts the hidden state $h^{<t>}$ should carry.

$$h^{<t>} = o^{<t>} * tanh(C^{<t>}) \tag{6}$$

The new $C^{<t>}$ and the new $h^{<t>}$ is then transferred to the next time step.

B. CSA

CSA [31] are a class of algorithms that belongs to the field of artificial immune systems (AIS). AIS are one of the youngest and most growing fields in nature-inspired computing [32]. AIS are computational intelligence techniques derived from the natural immune system functions and principles for providing protection and maintenance of our bodies [33]. The distinct features of the immune system, such as self-organization, learning through experience and memory, adaptation, recognition, robustness, and scalability, were the motives for developing new algorithms and systems to solve complex science and engineering problems [34], [35]. AIS emerged in the early 1990s [36], [37] on the basis of a proposal in the late 1980s to implement theoretical immunological models to machine learning and automated problem-solving [38], [39]. Early work in the field was inspired by immune network theory and applied to machine learning, optimization, and control problems.

Forrest *et al.* [40], [41] and Kephart *et al.* [42] introduced the immune system as an analogy for the development of applications in computer security. These efforts were developmental for the field of AIS because they created an intuitive field of application that fascinated the public and helped differentiate the work as an independent subdomain. Modern AIS are inspired by one of three main immunological theories: clonal selection, negative selection and immune network. The methods are commonly used for optimization, clustering, classification, anomaly detection, computer security and other similar machine learning problem domains [43].

CSA was inspired by Burnet's proposed clonal selection theory of acquired immunity [44], [45]. The theory explains how B and T lymphocytes improve their response to antigens over time, which is called affinity maturation. When a lymphocyte is selected and binds to an antigen determinant, the cell replicates, making many copies of itself and differentiating into different types of cells (plasma and memory cells). Then the selected cell undergoes somatic hypermutation that alters the shape of the expressed receptors and the capabilities to recognize subsequent determinants of both lymphocyte surface-bound antibodies and the antibodies produced by plasma cells [46].

CSA has shown to be robust in avoiding local minima, self-tuning (in terms of resources used), insensitive to user parameters and very competitive to solve search and optimization problems [21]. The algorithm uses analogs of a genetic representation (e.g., decimals or binary), affinity (function evaluations), bio-operators such as selection, cloning and affinity maturation (mutation) [47].

Algorithm 1 A General Pseudocode of the CSA

```
Data: Initial Antigen ($Ag$), Population Size ($N$) $\beta$, $\alpha$
Result: Best
while ¬StopCondition() do
    Population ← Create Initial Population ($N$);
    foreach $Ab \in$ Population do
        Affinity($Ab$);
    Selected Population ← Select(Population, $m$);
    Clones ← ∅;
    foreach $Ab \in$ Selected Population do
        Clones ← Clone($Ab$, $\beta$);
    Mutated Clones ← ∅;
    foreach $Ab \in$ Mutated Clones do
        Mutate($Ab$, $\alpha$);
        Affinity($Ab$);
    Best ← Select(Mutated Clones, Best Clone);
    Replace($Ag$, Best);
```

Algorithm 1 provides a pseudocode listing of the CSA. The CSA model involves first initialize a random potential solution called an antigen. In response to the antigen, a population of antibodies (candidate solutions) is randomly generated. The affinity score of each antibody is scored based on the predefined objective function. The antibodies that have a higher affinity (or fitness) score than the antigen will be selected. The selected antibodies m are subjected to cloning proportional to affinity (rank-based). The number of clones generated from each antibody is calculated as follows:

$$N_c = \text{round}\left(\frac{\beta \cdot N}{i}\right) \tag{7}$$

In (7), N_c is the total number of clones created for a given antibody, β is a clonal factor, N is the size of the population, i is the antibody current rank where $i \in [1, m]$, and round (.) is the operator that rounds its argument towards the closest integer.

The clones then undergo affinity maturation (mutation) inversely-proportional to clone affinity. The mutation rate can be implemented based on rules like the one defined in (8) [48]:

$$\alpha = \frac{1}{\rho}\exp(-f) \tag{8}$$

where α is the mutation rate, ρ controls its decay, and f is the antibody affinity normalized in [0,1].

The new affinity value of each clone is calculated, and the best clone is selected, and the rest of the clones are removed. Finally, the selected clone will evolve and becomes the primary antigen of the next generation, and the process is repeated for several iterations until the desired solution is found.

TABLE 1. Hyperparameters of the LSTM topology and their range.

Hyperparameter	Range
Number of Epochs	[1-16]
Batch Size	[16, 32, 64, 96, 128, 196, 256, 500]
Embedding vector size	[16, 32, 48, 64, 96, 128]
Number of LSTM layers	[1-3]
Number of LSTM units in each LSTM layer	[8, 16, 24, 32, 48, 64, 96]
Number of dense layers	[0-3]
Number of neurons for each dense layer (if any)	[1-49]
Optimizer	[SGD (1), Adadelta (2), RMSprop (3), Adam (4), Nadam (5)]
Activation function at the output layer	[Sigmoid (1), Softmax (2)]

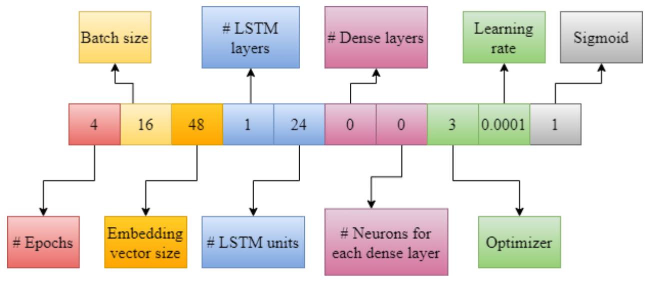

FIGURE 2. Integer encoding of the LSTM hyperparameters into an antigen.

IV. PROPOSED METHOD

In this section, we implement a framework using CSA to automatically design LSTM topologies to improve their performance to solve text classification problems. This involves the ability to evolve many hyperparameters of the topology. CSA can provide a powerful heuristic search for complex and large spaces if an efficient encoding of that search space is adopted. The encoding must enable the direct use of the mutation operators. The affinity evaluation method of CSA must also be determined and implemented. Given these constraints, we propose an integer encoding scheme for the CSA method. The integer encoding scheme can efficiently encode all the hyperparameters and possible LSTM topologies and enable different mutation applications. The hyperparameters associated with the LSTM topology and their ranges considered in our proposed approach are presented in Table 1.

Figure 4 shows the overall process to find the best hyperparameters for the LSTM model using CSA. The following steps IV-A through IV-H describe the complete procedure in detail.

A. ANTIGEN ENCODING

To use clonal selection algorithms, we must first represent the problem domain as an antigen. Here, we want to find an optimal set of hyperparameters for the LSTM model. Initial hyperparameters in the LSTM network are chosen randomly within their range specified in Table 1. The hyperparameters set can be represented by a 1D vector in which an integer number corresponds to a particular hyperparameter. In total, there are 10 hyperparameters. Since an antigen is a collection of genes, a set of hyperparameters can be represented by a 10-gene antigen, where each gene corresponds to a single hyperparameter. Figure 2 represents a random LSTM topology encoded into an antigen in integer format.

Below is a brief description of the topology hyperparameters to be optimized by CSA.

The first hyperparameter is the number of epochs. It defines the number of times the learning algorithm will run through the entire training data. Little training (very few epochs) will indicate that the model will underfit the training and the testing data. Much training (too many epochs) will indicate that the model will overfit the training dataset and perform poorly on the testing dataset. The batch size is an important hyperparameter that determines the number of examples propagated across the network before updating the model parameters (e,g. weights and biases). Generally, a larger batch size leads to faster training, but it requires more memory space and does not always converge as fast. Smaller batch size is slower to train, but it can converge faster. The embedding vector size is the dimensionality for each word vector. This hyperparameter belongs to the embedding layer, a technique for representing words using a dense vector representation. The embedding layer is always the first hidden layer of the LSTM network. LSTM layers and their number of units (LSTM units) are also critical hyperparameters to find their optimal values as they greatly impact the overall performance of the LSTM model. The possibility of adding dense layers and their optimal number of neurons are also included. Additionally, Relu is used as the activation function in dense layers (if there are any). Optimizer is likewise an important hyperparameter to be tuned/optimized. Optimizers are algorithms or techniques used to tweak and change the neural network's parameters, such as weights and biases, to minimize the losses. Finally, because we evaluate our approach with binary classification datasets (e.g., positive/negative and/or spam/not spam), the activation function used at the last (or output) layer is also optimized. If sigmoid is selected, then one neuron is used at the last layer with binary cross-entropy being the loss function, whereas if softmax is chosen, then two neurons are used, which is corresponds to the number of classes or labels of the dataset with categorical cross-entropy being the loss function.

B. ANTIBODIES CREATION

As the immune system creates several antibodies in response to fight the antigen, similarly, in the CSA algorithm, antibodies are created in response to the initial antigen initialized above. In simple terms, a random population of N antibodies (the number of LSTM networks with different hyperparameters) is generated as shown in Figure 3. Each antibody in the population is encoded in the same way as the antigen. For this work, the size of the population is selected to be 20.

C. AFFINITY EVALUATION

To evaluate each antibody's affinity, we need first to define the fitness function. Since this is a binary classification task, the test classification accuracy is used as the fitness function to be optimized or maximized. This means training the LSTM model weights and biases using a particular set of

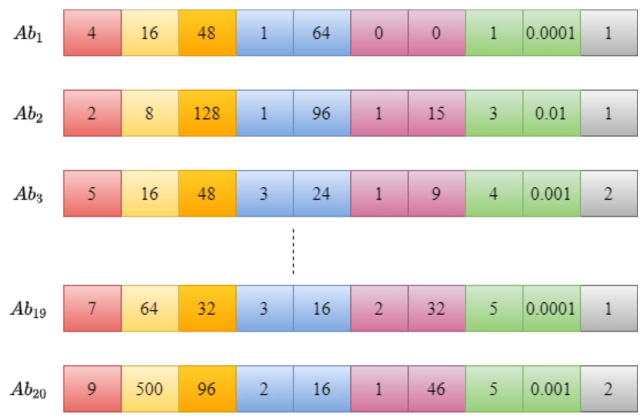
Ab_1 4 16 48 1 64 0 0 1 0.0001 1
Ab_2 2 8 128 1 96 1 15 3 0.01 1
Ab_3 5 16 48 3 24 1 9 4 0.001 2
Ab_{19} 7 64 32 3 16 2 32 5 0.0001 1
Ab_{20} 9 500 96 2 16 1 46 5 0.001 2

FIGURE 3. Initial population of candidate solutions.

hyperparameters determined by the antibody genes on the training dataset and then see how well it performs at classifying the test set. The higher the accuracy, the fitter the antibody. In other words, the CSA attempts to discover the best set of hyperparameters that maximize accuracy.

D. SELECTION

In the selection process, the antibodies with higher affinity than the antigen are selected to move forward for cloning and mutation stages. Antibodies with lower affinity than the antigen will not go forward for cloning and mutation, and they will be removed from the population. This is a good approach to reduce the execution time of the algorithm.

E. CLONING

In the cloning process, a number of clones will be generated from each of the selected antibodies. In this implementation, all selected antibodies, regardless of their affinity values, have the same clone size, which is fixed to 10 clones.

F. MUTATION

Mutation is the process of randomly altering genes in a given antibody to introduce diversity in the population. Mutation is a major part of the CSA, which enables to achieve global optimization and thus helping to escape from local optimization. In CSA, the mutation is inversely proportional to the affinity of an antibody. Meaning the higher the affinity, the lower the mutation and vice-versa. This is known as affinity maturation. For this work, mutation function is performed by selecting random genes in each clone (inversely proportional to the affinity with $\rho = 1.0$) and replacing their values with random values drawn from a truncated Gaussian distribution. The truncated Gaussian Distribution is defined in the same manner as the normal distribution: by the mean and standard deviation, and we also determine a range to limit the distribution to an upper, lower or double truncated distribution. Figure 5 represents a mutation process using the truncated Gaussian applied to a clone.

G. AFFINITY EVALUATION OF THE MUTATED CLONES

For each mutated clone, train the LSTM using the training dataset and calculate the accuracy (affinity) of LSTM model

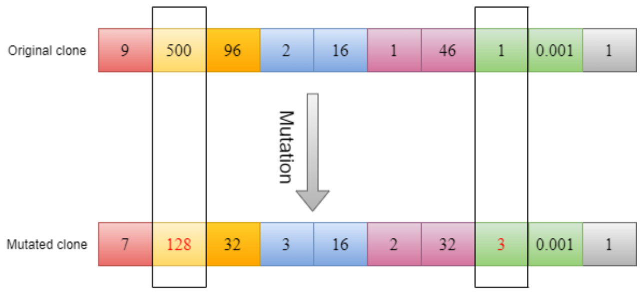
Original clone 9 500 96 2 16 1 46 1 0.001 1
Mutation
Mutated clone 7 128 32 3 16 2 32 3 0.001 1

FIGURE 5. Mutation operation for LSTM hyperparameters optimization.

using the test dataset. Once the affinity is calculated for all, we sort them in ascending order based on their affinity, then we select the one with the highest affinity and kills off the rest.

H. STOPPING CRITERION

The best clone replaces the original antigen and becomes the primary antigen for the next generation. The whole process is repeated multiple times until stopping criterion such as optimal solution (LSTM topology) or the maximum number of iterations (generations), which is 50 has reached.

V. EXPERIMENT DESIGN

A. DATASETS

Three widely used benchmark datasets for text classification are utilized to evaluate the performance of the proposed method and other algorithms. Each dataset is briefly described below.

1) IMDB DATASET

The large movie review dataset (IMDB) is a binary sentiment analysis classification dataset. The data was collected by Stanford researchers in 2011 [49]. The task is to classify the sentiment of a particular movie review as either positive or negative. The dataset includes 25,000 positive reviews and 25,000 negative reviews. The average movie review is below 300 words, with a standard deviation above 200 words.

2) TWITTER US AIRLINE DATASET

The dataset initially released by Crowdflower's Data for Everyone library comprises several tweets taken from the standard Kaggle Dataset: Twitter US Airline Sentiment [50]. A total of 14,640 tweets were extracted, which created the experimental dataset. The tweets collected were for the six major U.S. airlines of February 2015. Contributors were requested first to classify positive, negative, and neutral tweets, followed by describing negative reasons (such as ''late flight'' or ''rude service''). For this work, tweets are classified as either negative or positive; therefore, rows/samples with neutral sentiment were filtered out. After this, the dataset includes 2,363 positive tweets and 9,178 negative tweets. The task is to predict the sentiment of a tweet about US airlines as either positive or negative. The average tweet is below 110 words, with a standard deviation of just over 40 words.

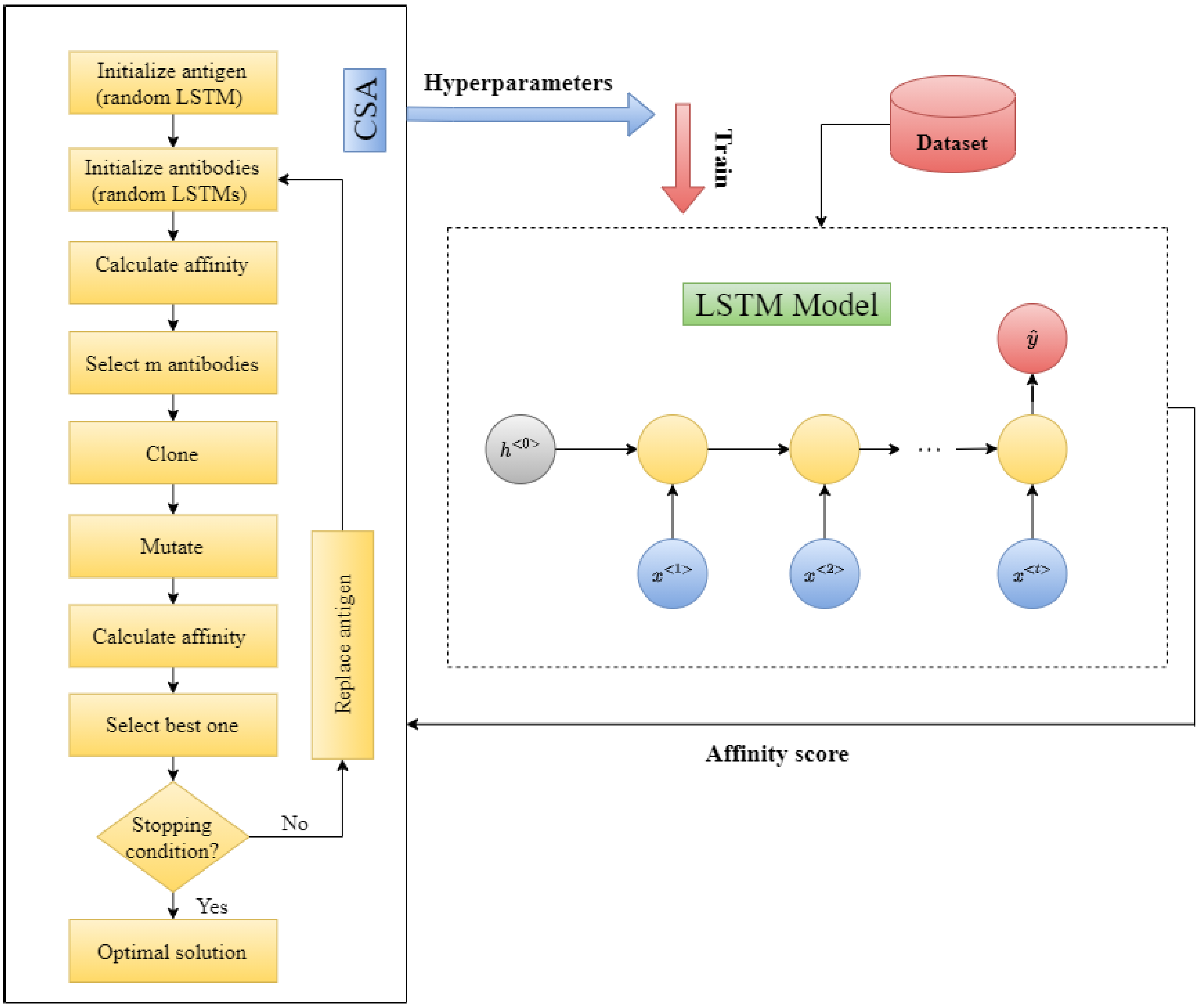

FIGURE 4. CSA-LSTM overall procedure.

3) SMS SPAM COLLECTION DATASET

The dataset is a public set of SMS-tagged messages that have been collected for SMS spam research and have been labeled as either spam or ham [51], [52]. It contains 5,574 SMS phone messages and labels. It is an imbalanced dataset with 4,825 ham compared to 747 spam messages, making it more challenging. The task is to classify a text message as spam or not (aka ham). The average SMS message is below 100 words, with a standard deviation of just over 100 words.

B. TEXT CLEANING AND PREPROCESSING

Machine learning and deep learning models require numeric data as they work best when inputs are numerical. Online textual data such as reviews and tweets are usually inconsistent and lack specific features or missing values that must be handled before any analysis is conducted. Encoding techniques like TF-IDF, BagOfWord, and Word2Vec are often used to transform the text data into a numeric vector. However, before encoding, the textual data must be cleaned up first. The following cleaning steps have been done for all three datasets:

- **Data filtration:** So that it contains the only data required for sentiment-based classification. For instance, the data columns needed for the Twitter US Airline Sentiment dataset are the airline sentiment and text columns. It's a classification problem; thus, that text will be the features, and airline sentiment will be the labels.
- **Data shuffle:** Random shuffle of the data before splitting it between the train and the test sets to ensure that the sentiment classes are distributed evenly across the train and test sets.
- **Train-test data split:** The performance evaluation of the model should be performed on a test set. By doing so, we can evaluate how robust the generalization of the model is.
- **Removing irrelevant words:** Remove all unnecessary words that do not add any meaning to the sentence

(URLs, HTML tags, numbers, punctuation, etc.), and all characters are converted into lowercase to avoid duplication.

- **Tokenization:** Each given text was split into chunks of words.
- **Removing stopwords:** Stopwords like "a," "the," "is," "all," etc., that do not carry significant meaning were removed from texts.
- **Normalization:** All words were normalized using the lemmatization approach.
- **Mapping words to integers:** Every single word or token was mapped into a unique integer
- **Padding/truncating:** All (inputs) must have the same length before feeding them to the model. For example, if the maximum input length is limited to 500, inputs longer than 500 will be truncating, and inputs that are shorter than 500 are padded with zeros.
- **Word embedding:** Discrete words are encoded as real-valued vectors in a high-dimensional space for consumption by the deep learning model. There are many word embedding techniques; for this research, we have used the Keras embedding layer, which provides an easy means to transform positive integer representations of words into word embeddings.

C. PERFORMANCE METRICS

Selecting the proper metric is crucial while evaluating deep and machine learning models' performance. Several metrics have been proposed for evaluating machine learning models in different applications. In some applications, especially when dealing with imbalanced data looking at a particular metric may not provide the full picture of the problem being solved, and we may need to apply a subset of the metrics to get a concrete evaluation of the models. For this reason, we used well-known metrics such as accuracy, precision, recall, and F1-score for performance evaluation [53]. Before describing the metrics, there are four major terms to define:

- **True Positives (TP):** The instances in which the model correctly predicts positive and the true output is also positive.
- **True Negatives (TN):** The instances in which the model correctly predicts negative and the true output is also negative.
- **False Positives (FP):** The instances in which the model predicts positive and the true output is negative.
- **False Negatives (FN):** The instances in which the model predicts negative and the true output is positive.

1) Accuracy: It is a good indicator of the model performance when the class distribution is balanced. It is calculated as the ratio between the total number of correct predictions to the total number of predictions.

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

2) Precision: It is a good indicator of the model performance when the class distribution is imbalanced. Precision is calculated as the ratio between the total number of positive examples correctly classified to the total number of examples classified correctly or incorrectly as positive. In other words, The precision metric measures the accuracy of the model in classifying an example as positive.

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

3) Recall: It can be defined as the ratio between the number of correctly classified positive examples to the total number of positive samples. In other words, the recall metric measures the model's ability to recognize positive examples. The higher the recall, the more positive examples recognized.

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

4) F1-score: It helps to have a measurement that represents both recall and precision. It is calculated as the harmonic mean of precision and recall.

$$\text{F1-score} = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{12}$$

1) EXPERIMENTAL SETUP

The CSA-LSTM was implemented in Python 3.7 using a high-level neural networks API called Keras running on top of TensorFlow. The research was conducted with a Lenovo laptop powered by a processor-Intel(R) Core(TM) i7-9750HX CPU @ 2.60GHz and 32 GB DDR4 RAM with Nvidia GEFORCE GTX GPU with 6 GB of GDDR5 memory. The estimated work time of the CSA to find an optimal set of hyperparameters for the LSTM model was approximately 1.5 hours.

The classification process using LSTM is illustrated in Figure 6. First, the text data was clean and preprocessed; afterward, word embedding, particularly the embedding layer, was used to transform the textual data into numeric vectors (feature extraction). After that, the training is done using the LSTM model with the optimal hyperparameters discovered by the CSA. Lastly, a prediction is made on test data, and the performance is evaluated based on the metrics described above.

The proposed CSA-LSTM was benchmarked using the IMDB dataset. The obtained optimal hyperparameters was also used for two other datasets: Twitter US Airline Sentiment and the SMS spam collection. However, The Epochs hyperparameter, which CSA found to be 4, was changed to 10 for the Twitter US Airline dataset and 14 for the SMS Spam dataset. Another possible choice to avoid increasing the number of epochs is to reduce the batch size from 500 (found by CSA) to another size like 128 or 64.

Table 2 shows each dataset's specification, and Table 3 provides the final optimized hyperparameters of the LSTM topology optimized by the CSA.

The LSTM hyperparameters found by CSA were also combined with pre-determined CNN layers as shown in Figure 7.

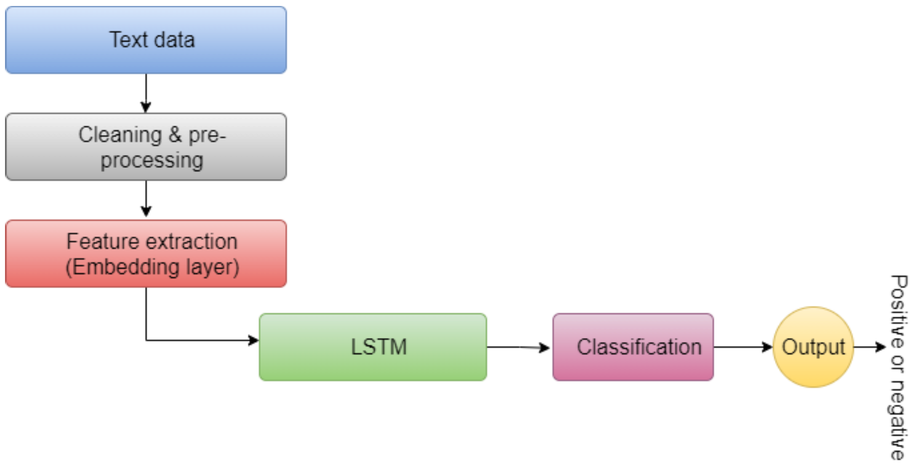
Text data
Cleaning & pre-processing
Feature extraction (Embedding layer)
LSTM
Classification
Output
Positive or negative

FIGURE 6. A diagram of the text classification process using LSTM model.

TABLE 2. Specification of the IMDB, SMS Spam, and Twitter US Airline datasets.

	IMDB	SMS Spam	Twitter US Airline
Input length	500	100	200
Vocabulary size	181,556	8,024	13,234
#Training examples	45,000	4,457	9,232
#Test examples	5,000	1,115	2,309
#Classes	2	2	2

TABLE 3. The final optimal hyperparameters of the LSTM topology discovered by our proposed CSA method.

Hyperparameter	Value
#Epochs	4
Batch size	500
Embedding vector size	16
#LSTM layers	1
#LSTM units	32
#Dense layers	0
#Neurons for each dense layer	0
Optimizer	Adam ($\eta = 0.01$)
Activation function at the output layer	Sigmoid

This technique is referred to as CSA-CNN-LSTM in this research. It attempts to deliver similar or better results to the first (CSA-LSTM) to fast the training time and less parameters.

For further evaluation, the research also implemented four other machine learning algorithms widely used for text classification tasks, namely random forest, logistic regression, support vector machine, and multinomial naive Bayes. For these four algorithms, extracting features from the text data was done by applying the bag-of-words (BOW) and the TF-IDF vectorization techniques. These features or vectors are then fed directly as inputs into the machine-learning algorithms. Both vectorization methods will produce different numbers of features for each text or review, depending on the availability of the number of tokens in the text. As described earlier, this problem can be solved by using padding. We have observed that these machine learning algorithms achieved higher accuracy using the TF-IDF than the BOW approach. Therefore, we included only the algorithms' results using the TF-IDF approach and compare their performance with the research's proposed models. We implemented the four algorithms with the help of the scikit-learn library. Additionally, We have not customized the algorithm's hyperparameters; rather, we chose the sensible default values for the hyperparameters of each model provided in the scikit-learn library.

TABLE 4. Number of parameters, GPU training time, and test accuracy of our proposed CSA-LSTM model.

Dataset	#Parameters	Training time (sec)	Test accuracy
IMDB	166,305	83.2	89.52%
SMS Spam	134,689	4.4	98.48%
Twitter US Airline	218,049	5.6	92.25%

TABLE 5. Number of parameters, GPU training time, and test accuracy of our proposed CSA-CNN-LSTM model.

Dataset	#Parameters	Training time (sec)	Test accuracy
IMDB	165,673	11.3	89.88%
SMS Spam	134,057	3.4	98.74%
Twitter US Airline	217,417	4.8	92.77%

VI. RESULTS

The performance of our proposed approaches CSA-LSTM and CSA-CNN-LSTM in terms of the number of trainable parameters, GPU training time (sec), and test accuracy for the three datasets is provided in Tables 4 and 5, respectively. We can see that the CSA-CNN-LSTM achieves better accuracy than CSA-LSTM, although with fewer parameters and faster training time. This validated our proposed hypothesis that combining CNN with LSTM reduces the training time and often leads to better accuracy.

A. PERFORMANCE COMPARISON

1) COMPARISON WITH OUR MACHINE LEARNING ALGORITHMS

Table 6 compares the performance of the proposed models against the four machine learning algorithms that are evaluated on the IMDB dataset. It shows that the performance of CSA-CNN-LSTM outperforms all other models in all evaluation metrics followed by the CSA-LSTM.

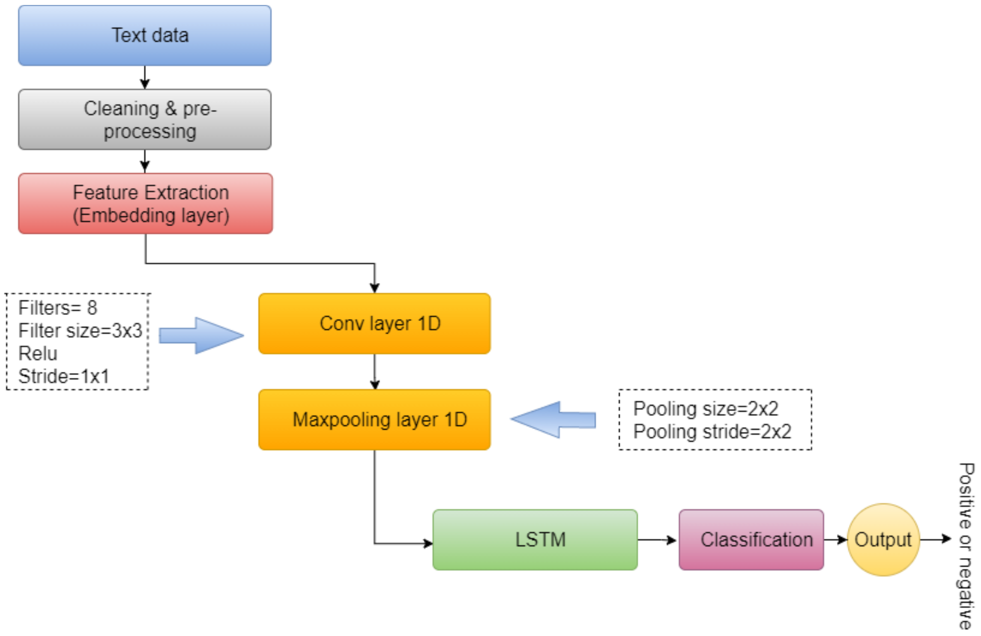

FIGURE 7. A diagram of the text classification process using CSA-CNN-LSTM model.

TABLE 6. Performance evaluation comparison for all models on the IMDB dataset.

Model	Accuracy	Precision	Recall	F1- score
CSA-LSTM	89.52%	88.58%	90.77%	89.66%
CSA-CNN-LSTM	89.88%	89.37%	90.05%	89.71%
Random forest	54.66%	54.38%	52.34%	53.34%
Logistic regression	51.60%	51.22%	47.41%	49.24%
Support vector machine	53.04%	52.37%	57.10%	54.63%
Multinomial naive bayes	50.12%	49.78%	85.50%	62.93%

TABLE 7. Performance evaluation comparison for all models on the SMS Spam dataset.

Model	Accuracy	Precision	Recall	F1-score
CSA-LSTM	98.48%	97.05%	91.03%	93.95%
CSA-CNN-LSTM	98.74%	93.33%	96.55%	94.91%
Random forest	92.64%	78.89%	59.31%	67.71%
Logistic regression	85.02%	33.33%	15.17%	20.85%
Support vector machine	89.05%	67.69%	30.34%	41.90%
Multinomial naive bayes	80.35%	36.39%	68.27%	47.48%

The results of all models for the SMS Spam dataset are provided in Table 7. CSA-CNN-LSTM again delivers the best results, followed by CSA-LSTM. The SMS dataset is too small and has an imbalanced number of instances for each class. The performance could be further improved using data augmentation techniques such as the Synthetic Minority Oversampling technique or SMOTE for short [54]. This simple method includes duplicating instances in the minority class.

In Table 8, the results of the models for the Twitter US Airline dataset are given. This dataset also has an imbalanced number of instances for each class, and the performance considerably could be enhanced using the SMOTE approach. CSA-CNN-LSTM again outperforms other methods, followed by CSA-LSTM. Once more, these results evidence the proposed CSA algorithm's competence in designing optimal LSTM topologies that provide the best performance for NLP problems.

TABLE 8. Performance evaluation comparison for all models on the Twitter US Airline dataset.

Model	Accuracy	Precision	Recall	F1-score
CSA-LSTM	92.25%	94.06%	95.94%	94.99%
CSA-CNN-LSTM	92.77%	94.50%	95.88%	95.19%
Random forest	83.32%	83.85%	89.05%	90.40%
Logistic regression	80.03%	80.06%	99.94%	88.91%
Support vector machine	80.07%	80.10%	99.94%	88.93%
Multinomial naive bayes	53.44%	90.31%	46.89%	61.73%

TABLE 9. Comparative accuracy against other methods.

Model	Dataset	Accuracy
CNN-LSTM [55]	IMDB	88.90%
Bag of Words [49]	IMDB	88.89%
Doc2VecC [56]	IMDB	88.30%
coRNN [57]	IMDB	87.4%
S-LSTM [58]	IMDB	87.15%
Reverse DR-AGG [59]	IMDB	45.5%
Standard DR-AGG [59]	IMDB	45.1%
RNN [60]	SMS Spam	98.11%
SVM-based spam filter [61]	SMS Spam	97.81%
NB-based spam filter [61]	SMS Spam	80.54%
SVM + tok1 [62]	SMS Spam	97.64%
AdaBoost [63]	Twitter US Airline	84.5%

2) COMPARISON AGAINST OTHER MODELS IN THE LITERATURE

Table 9 compares our proposed models' accuracy against other models that have been evaluated on the same three datasets.

For the IMDB dataset, the best accuracy among the existing research works [55] using manually designed CNN-LSTM is 88.90%. Our proposed CSA-LSTM model shows an accuracy of 89.52%. Our CSA-LSTM model on the SMS Spam dataset outperform other models implemented using RNN, SVM, NB and SVM + tok1 with an accuracy of 98.48%. The CSA-LSTM model shows an accuracy of 92.25% with a margin of 7% increase compared to the existing work on the Twitter US Airline dataset. The results prove that our proposed model is consistently better compared to other models.

VII. CONCLUSION AND FUTURE WORK

Designing LSTM-based RNN topologies is a very challenging task due to the many hyperparameters that need to be configured. With the large number of hyperparameters for most LSTM models nowadays, it is not easy to manually find a desirable configuration for a specific task. Due to the shortcomings of current methods and the limited computing resources available to experimenters, optimizing LSTM topologies are often performed by domain experts who adopt innovative theoretical insights and intuitions gained from experience. This research has successfully implemented a novel framework for the automatic design of LSTM topologies using CSA to address text classification tasks. The proposed framework aims to help the deep learning community design optimal LSTM topologies in a fully automatic manner without the need for expert knowledge or much trial and error. Our CSA approach features the use of a simple encoding scheme for encoding LSTM topologies in integer vector representation. We have also implemented an efficient mutation scheme using Truncated Gaussian. Truncated Gaussian mutation was found to be efficient and fast in finding a solution near optimum. We have also introduced an affinity evaluation process to measure each evolved topology fitness quickly and accurately. Our experiments were carried out on three challenging benchmark datasets. Experimental results show that our proposal outperforms nearly all the state-of-the-art machine learning methods used for text classification problems and many other models reported in the literature. Additionally, the optimal hyperparameters of LSTM topology found by CSA were combined with pre-determined CNN layers to improve the CSA-LSTM performance with less weights and faster training time. With the help of CSA, we found cost-efficient, less complex, and reusable topology that can work with multiple datasets and still achieve high accuracy. We also saw how the problem of overfitting could be avoided without using any regularization method if the right number of hyperparameters and their values are used, which was fulfilled through our CSA method. Therefore, this research signifies that CSA is a powerful technique that should be considered as an alternative to other techniques to automatically optimize LSTM model hyperparameters and search for the ideal topology.

The future direction of the current research would focus on extensive evaluations of the CSA method on a large number of benchmark datasets. Another potential future direction in this domain would be to propose efficient CSA methods to design different LSTM topologies automatically to solve important NLP tasks such as automatic summarization, machine translation, speech recognition, and image captioning.

REFERENCES

[1] A. S. A. Bataineh, "A gradient boosting regression based approach for energy consumption prediction in buildings," *Adv. Energy Res.*, vol. 6, no. 2, pp. 91–101, 2019.

[2] A. A. Bataineh, A. Mairaj, and D. Kaur, "Autoencoder based semi-supervised anomaly detection in turbofan engines," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 11, 2020, doi: 10.14569/IJACSA.2020.0111105.

[3] I. Goodfelow, Y. Bengio, and A. Courville, *Deep Learning* (Adaptive Computation and Machine Learning Series). 2016.

[4] S. M. J. Jalali, S. Ahmadian, A. Khosravi, S. Mirjalili, M. R. Mahmoudi, and S. Nahavandi, "Neuroevolution-based autonomous robot navigation: A comparative study," *Cognit. Syst. Res.*, vol. 62, pp. 35–43, Aug. 2020.

[5] S. M. J. Jalali, P. M. Kebria, A. Khosravi, K. Saleh, D. Nahavandi, and S. Nahavandi, "Optimal autonomous driving through deep imitation learning and neuroevolution," in *Proc. IEEE Int. Conf. Syst., Man Cybern. (SMC)*, Oct. 2019, pp. 1215–1220.

[6] S. M. J. Jalali, S. Ahmadian, M. Khodayar, A. Khosravi, V. Ghasemi, M. Shafie-Khah, S. Nahavandi, and J. P. S. Catalão, "Towards novel deep neuroevolution models: Chaotic levy grasshopper optimization for short-term wind speed forecasting," *Eng. Comput.*, pp. 1–25, Mar. 2021.

[7] A. A. Bataineh and D. Kaur, "A comparative study of different curve fitting algorithms in artificial neural network using housing dataset," in *Proc. IEEE Nat. Aerosp. Electron. Conf. (NAECON)*, Jul. 2018, pp. 174–178.

[8] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Mar. 2013.

[9] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2673–2681, Nov. 1997.

[10] R. Socher, C. C.-Y. Lin, A. Y. Ng, and C. D. Manning, "Parsing natural scenes and natural language with recursive neural networks," in *Proc. ICML*, 2011.

[11] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," 2015, *arXiv:1508.07909*. [Online]. Available: http://arxiv.org/abs/1508.07909

[12] B. Liu, "Sentiment analysis and opinion mining," *Synthesis Lectures Hum. Lang. Technol.*, vol. 5, no. 1, pp. 1–167, 2012.

[13] J. Mao, W. Xu, Y. Yang, J. Wang, Z. Huang, and A. Yuille, "Deep captioning with multimodal recurrent neural networks (m-RNN)," 2014, *arXiv:1412.6632*. [Online]. Available: http://arxiv.org/abs/1412.6632

[14] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 6645–6649.

[15] A. A. Bataineh and D. Kaur, "Optimal convolutional neural network architecture design using clonal selection algorithm," *Int. J. Mach. Learn. Comput.*, vol. 9, no. 6, pp. 788–794, Dec. 2019.

[16] J. Brownlee, "Machine learning mastery with Python," *Mach. Learn. Mastery Pty Ltd*, vol. 527, pp. 100–120, 2016.

[17] S. M. J. Jalali, S. Ahmadian, A. Khosravi, M. Shafie-Khah, S. Nahavandi, and J. P. S. Catalao, "A novel evolutionary-based deep convolutional neural network model for intelligent load forecasting," *IEEE Trans. Ind. Informat.*, early access, Mar. 12, 2021, doi: 10.1109/TII.2021.3065718.

[18] A. A. Bataineh and D. Kaur, "Immuno-computing-based neural learning for data classification," *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 6, 2019, doi: 10.14569/IJACSA.2019.0100632.

[19] I. Muthreja and D. Kaur, "A comparative analysis of immune system inspired algorithms for traveling salesman problem," in *Proc. Int. Conf. Artif. Intell. (ICAI), Steering Committee World Congr. Comput. Sci., Comput.*, 2018, pp. 164–170.

[20] J. Brownlee, "Clonal selection theory & clonalg-the clonal selection classification algorithm (CSCA)," Swinburne Univ. Technol., Melbourne, VIC, Australia, Tech. Rep., 2005, p. 38.

[21] J. Brownlee, "Clonal selection algorithms," Complex Intell. Syst. Lab., Swinburne Univ. Technol., Melbourne, VIC, Australia, Tech. Rep., 2007.

[22] H. Chung and K.-S. Shin, "Genetic algorithm-optimized long short-term memory network for stock market prediction," *Sustainability*, vol. 10, no. 10, p. 3765, Oct. 2018.

[23] S. Stajkowski, D. Kumar, P. Samui, H. Bonakdari, and B. Gharabaghi, "Genetic-algorithm-optimized sequential model for water temperature prediction," *Sustainability*, vol. 12, no. 13, p. 5374, Jul. 2020.

[24] V. C. L. Neto, L. A. Passos, and J. P. Papa, "Evolving long short-term memory networks," in *Proc. Int. Conf. Comput. Sci.* Springer, 2020, pp. 337–350.

[25] D. Chhachhiya, A. Sharma, and M. Gupta, "Designing optimal architecture of recurrent neural network (LSTM) with particle swarm optimization technique specifically for educational dataset," *Int. J. Inf. Technol.*, vol. 11, no. 1, pp. 159–163, Mar. 2019.
[26] B. Nakisa, M. N. Rastgoo, A. Rakotonirainy, F. Maire, and V. Chandran, "Long short term memory hyperparameter optimization for a neural network based emotion recognition framework," *IEEE Access*, vol. 6, pp. 49325–49338, 2018.
[27] A. ElSaid, F. El Jamiy, J. Higgins, B. Wild, and T. Desell, "Optimizing long short-term memory recurrent neural networks using ant colony optimization to predict turbine engine vibration," *Appl. Soft Comput.*, vol. 73, pp. 969–991, Dec. 2018.
[28] A. D. Yuliyono and A. S. Girsang, "Artificial bee colony-optimized LSTM for bitcoin price prediction," *Adv. Sci., Technol. Eng. Syst. J.*, vol. 4, no. 5, pp. 375–383, 2019.
[29] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[30] C. Olah, "Understanding LSTM networks," Tech. Rep., 2015.
[31] L. N. De Castro and F. J. Von Zuben, "The clonal selection algorithm with engineering applications," in *Proc. GECCO*, 2000, pp. 36–39.
[32] L. N. D. Castro and J. I. Timmis, "Artificial immune systems as a novel soft computing paradigm," *Soft Comput. A, Fusion Found., Methodol. Appl.*, vol. 7, no. 8, pp. 526–544, Aug. 2003.
[33] J. Brownlee, "Artificial immune recognition system (AIRS)—A review and analysis," Center Intell. Syst. Complex Processes (CISCP), Dept. Inf. Commun. Technol. (ICT), Swinburne Univ. Technol., Melbourne, VIC, Australia, Tech. Rep. 1-02, 2005.
[34] L. N. De Castro, "An introduction to the artificial immune systems," in *Proc. ICANNGA*, 2001.
[35] J. Timmis, A. Hone, T. Stibor, and E. Clark, "Theoretical advances in artificial immune systems," *Theor. Comput. Sci.*, vol. 403, no. 1, pp. 11–32, Aug. 2008.
[36] H. Bersini and F. J. Varela, "Hints for adaptive problem solving gleaned from immune networks," in *Proc. Int. Conf. Parallel Problem Solving Nature*. Springer, 1990, pp. 343–354.
[37] Y. Ishida, "Fully distributed diagnosis by PDP learning algorithm: Towards immune network PDP model," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 1990, pp. 777–782.
[38] J. D. Farmer, N. H. Packard, and A. S. Perelson, "The immune system, adaptation, and machine learning," *Phys. D, Nonlinear Phenomena*, vol. 22, nos. 1–3, pp. 187–204, Oct. 1986.
[39] G. W. Hoffmann, "A neural network model based on the analogy with the immune system," *J. Theor. Biol.*, vol. 122, no. 1, pp. 33–67, Sep. 1986.
[40] S. Forrest, S. A. Hofmeyr, and A. Somayaji, "Computer immunology," *Commun. ACM*, vol. 40, no. 10, pp. 88–96, 1997.
[41] S. Forrest, A. S. Perelson, L. Allen, and R. Cherukuri, "Self-nonself discrimination in a computer," in *Proc. IEEE Comput. Soc. Symp. Res. Secur. Privacy*, May 1994, pp. 202–212.
[42] J. O. Kephart, G. B. Sorkin, W. C. Arnold, D. M. Chess, G. J. Tesauro, S. R. White, and T. Watson, "Biologically inspired defenses against computer viruses," in *Proc. IJCAI (1)*, 1995, pp. 985–996.
[43] E. Hart and J. Timmis, "Application areas of AIS: The past, the present and the future," *Appl. Soft Comput.*, vol. 8, no. 1, pp. 191–201, Jan. 2008.
[44] F. M. Burnet, "A modification of Jerne's theory of antibody production using the concept of clonal selection," *Austral. J. Sci.*, vol. 20, no. 3, pp. 9–67, 1957.
[45] S. F. M. Burnet, *The Clonal Selection Theory of Acquired Immunity*, vol. 3. Nashville, TN, USA: Vanderbilt Univ. Press, 1959.
[46] J. Brownlee, *Clever Algorithms: Nature-Inspired Programming Recipes*. Jason Brownlee, 2011.
[47] L. N. de Castro and F. J. Von Zuben, "Learning and optimization using the clonal selection principle," *IEEE Trans. Evol. Comput.*, vol. 6, no. 3, pp. 239–251, Jun. 2002.
[48] V. Cutello, G. Narzisi, G. Nicosia, and M. Pavone, "Clonal selection algorithms: A comparative case study using effective mutation potentials," in *Proc. Int. Conf. Artif. Immune Syst.* Springer, 2005, pp. 13–28.
[49] A. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proc. 49th Annu. Meeting Assoc. Comput. Linguistics, Hum. Lang. Technol.*, 2011, pp. 142–150.
[50] U. Twitter. *Airline Sentiment*. Accessed: Aug. 11, 2019. [Online]. Available: https://www.kaggle.com/crowdflower/twitter-airline-sentiment
[51] T. A. Almeida, J. M. G. Hidalgo, and A. Yamakami, "Contributions to the study of SMS spam filtering: New collection and results," in *Proc. 11th ACM Symp. Document Eng. (DocEng)*, 2011, pp. 259–262.
[52] C. Tagg, "A corpus linguistics study of sms text messaging," Ph.D. dissertation, Univ. Birmingham, Birmingham, U.K., 2009.
[53] A. F. Gad, A. F. Gad, and S. John, *Practical Computer Vision Applications Using Deep Learning With CNNs*. Springer, 2018.
[54] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, Jun. 2002.
[55] J. Camacho-Collados and M. T. Pilehvar, "On the role of text preprocessing in neural network architectures: An evaluation study on text categorization and sentiment analysis," 2017, *arXiv:1707.01780*. [Online]. Available: http://arxiv.org/abs/1707.01780
[56] M. Chen, "Efficient vector representation for documents through corruption," 2017, *arXiv:1707.02377*. [Online]. Available: http://arxiv.org/abs/1707.02377
[57] T. K. Rusch and S. Mishra, "Coupled oscillatory recurrent neural network (coRNN): An accurate and (gradient) stable architecture for learning long time dependencies," 2020, *arXiv:2010.00951*. [Online]. Available: http://arxiv.org/abs/2010.00951
[58] Y. Zhang, Q. Liu, and L. Song, "Sentence-state LSTM for text representation," 2018, *arXiv:1805.02474*. [Online]. Available: http://arxiv.org/abs/1805.02474
[59] J. Gong, X. Qiu, S. Wang, and X. Huang, "Information aggregation via dynamic routing for sequence encoding," 2018, *arXiv:1806.01501*. [Online]. Available: http://arxiv.org/abs/1806.01501
[60] R. Taheri and R. Javidan, "Spam filtering in SMS using recurrent neural networks," in *Proc. Artif. Intell. Signal Process. Conf. (AISP)*, Oct. 2017, pp. 331–336.
[61] M. V. C. Aragão, E. P. Frigieri, C. A. Ynoguti, and A. P. Paiva, "Factorial design analysis applied to the performance of SMS anti-spam filtering systems," *Expert Syst. Appl.*, vol. 64, pp. 589–604, Dec. 2016.
[62] N. Al Moubayed, T. Breckon, P. Matthews, and A. S. McGough, "SMS spam filtering using probabilistic topic modelling and stacked denoising autoencoder," in *Proc. Int. Conf. Artif. Neural Netw.* Springer, 2016, pp. 423–430.
[63] A. Rane and A. Kumar, "Sentiment classification system of Twitter data for US airline service analysis," in *Proc. IEEE 42nd Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, vol. 1, Jul. 2018, pp. 769–773.

ALI Al BATAINEH (Member, IEEE) received the B.Sc. degree in computer engineering from Yarmouk University, Jordan, in 2010, and the M.Sc. degree in computer engineering from the University of Bridgeport, CT, USA, in 2016. He is currently pursuing the Ph.D. degree in electrical engineering with The University of Toledo, OH, USA. His research interests include deep learning, natural language processing, sentiment analysis, computer vision, metaheuristic optimization, and fuzzy logic.

DEVINDER KAUR (Life Senior Member, IEEE) received the B.Sc. and M.Sc. degrees (Hons.) in physics, majoring in electronics from Panjab University, in 1969 and 1970, respectively, the M.Sc. degree in medical physics from the University of Aberdeen, U.K., in 1976, and the M.Sc. and Ph.D. degrees in computer engineering from Wayne State University, USA, in 1985 and 1989, respectively. She was a Scientist with the Central Scientific Instruments Organization, a National Laboratory, Ministry of Science and Technology, Chandigarh, India, from 1971 to 1981. In 1989, she joined The University of Toledo as a Faculty Member, where she is currently a Full Professor with the Department of EECS. She has published upward of 100 articles in refereed journals and proceedings of the international conferences. She has worked on projects funded by NSF, Air Force Research Laboratory (AFRL), Daimler Chrysler, and ROMAN Engineering. Her research interests include developing intelligent applications based on hybrid computational models using biologically inspired computing and fuzzy systems. She was a recipient of the IIT Delhi Fellowship, from 1970 to 1971. She was the Fulbright Senior Specialist Award, in 2004, and visited the Nippon Institute of Technology Japan in that capacity. She was received the Commonwealth Scholarship Award for her M.Sc. degree.

• • •