IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 22, 2021, accepted June 14, 2021, date of publication June 23, 2021, date of current version July 2, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3091376

Arabic Text Classification Using Convolutional Neural Network and Genetic Algorithms

DEEM ALSALEH[1,2] AND SOUAD LARABI-MARIE-SAINTE[2]
[1]Saudi Information Technology Company (SITE), Riyadh 12382, Saudi Arabia
[2]College of Computer and Information Sciences, Prince Sultan University, Riyadh 11586, Saudi Arabia

Corresponding author: Souad Larabi-Marie-Sainte (slarabi@psu.edu.sa)

The authors would like to acknowledge the support of Prince Sultan University for paying the Article Processing Charges (APC) of this publication. This work was also supported by the Emerging Intelligent Autonomous Systems Data Science and Blockchain Lab (EIAS), Prince Sultan University, Riyadh, Saudi Arabia.

ABSTRACT Arabic documents are massively rising due to numerous contents utilized in websites, social media, and news articles. The classification of such documents in labelled categories is a significant and vital task that deserves more attention. Arabic Text Classification is an emerging research theme in Arabic Natural Language Processing. Recently, Deep Neural Network approaches have successfully been applied to many text classification problems, especially in English Text Classification. Convolutional Neural Network (CNN) is one of the best popular models. However, CNN is not highly applied in Arabic Text Classification. In addition, the recent studies did not achieve a high classification accuracy due to parameter setting issue. To overcome this limitation, a new hybrid classification model for Arabic Text is developed. This paper proposes Genetic Algorithms based Convolutional Neural Network for Arabic Text Classification. Genetic Algorithm is used to optimize the CNN parameters. The proposed model is tested using two large datasets and compared with the state-of-the art studies. The results showed that the classification accuracy achieved an improvement of 4 to 5%.

INDEX TERMS Natural language processing, Arabic text classification, convolutional neural networks, genetic algorithm.

I. INTRODUCTION

The Arabic language is the second hardest language and the fifth most used language around the world [1]. Even with the importance of Arabic language, there is still a gap in the Arabic Natural Language Processing (ANLP) due to its complexity and hardness. This complexity is raised through its diversified dialects, high derivative nature, and ambiguousness caused by diacritic [2]

ANLP is used in various fields such as Handwritten recognition [3], Named Entity Recognition [4], Information Extraction [5], Classification ([6]–[8]), Summarization, Translation [9], etc.

Arabic Text Classification is a crucial research area combining Arabic Natural Language Processing (ANLP) and Machine Learning (ML). It is about assigning documents to one or more classes based on their contents. Most of the existing research studies tackled English Text Classification using ML, while few are concerned with the Arabic text classification. This study presents a new classification model for Arabic text based on Deep Learning (DL). Deep Learning has attained exceptional expansions in numerous fields such that Computer Vision [6], healthcare [10] and it is progressively expanding in ANLP [11]. DL is based on Neural Network technique. Generally, the multiple processing layers in Deep Learning allow processing a huge amount of data more efficiently [12]. Convolutional Neural Network is one of most common DL algorithm that has successfully been applied to many domains including ANLP, for example in Text Classification ([6], [7]) and sentiment analysis ([13]–[15]). However, the obtained results were somewhat satisfactory due to the pre-processing phase [6] and the parameter setting of the DL methods. In fact, the gradient-descent used in CNN and other DL techniques might get stuck in the local optima due to the random initialization of the weigh values. These models yield best results when the weight' values are initially well set. For that, optimization algorithms can be used to find the most suitable parameters' value, especially the weight's values. This article proposes a new hybrid classification model for Arabic text based on CNN and Genetic Algorithm (GA). Genetic Algorithm is the most well-known optimization algorithm that proved its efficiency in many

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

fields. To the best of our knowledge, even though the optimization methods were proved to enhance the Deep Learning results ([16], [17]), it has not been yet applied for Arabic text classification. Hence, the contribution of this research is twofold. 1)- Propose a new hybrid Arabic text classifier based on Genetic Algorithm and Convolutional Neural Networks. 2)- Enhance the classification accuracy by optimizing the CNN weight vector using GA.

The proposed model is validated using two large datasets derived from the Modern Standard Arabic (MSA) format. The newly published Saudi Newspapers Articles Dataset (SNAD) [18] composed of 45935 documents, and the well-known Moroccan Newspapers Articles Dataset (MNAD) composed of 111728 documents. The proposed model is compared with a baseline, defined to be the Arabic text classification using CNN without parameter setting process, in addition to the state-of-the-art studies. The obtained results are validated using different classification metrics such as the accuracy, precision, recall, and F1 measure.

This study presents a pioneering hybrid method for Arabic text classification using GA based CNN. GA is used for hyperparameter tuning in the Convolutional Neural Networks. This combination is the first of its kind in Arabic Text Classification. Firstly, a deep preprocessing is performed to the datasets. For the data representation, the Glove technique is used. After that, the classification is employed using the combined method. The objectives of this research article are four:

- Investigate the limitations of the existing Arabic text classification techniques.
- Perform the most suitable pre-processing techniques on raw Arabic text to make it ready for classification.
- Propose a new model to improve the classification accuracy of Arabic text classification.
- Perform a comparison study to confirm the efficiency of the proposed model.

The rest of the article is organized as follows. Section II talked about the importance of the classification in NLP. Section III presents the recent related studies. Section IV introduces the main concepts utilized in this study such as GA, CNN, and ANLP. Section V describes the methodology followed to fulfill the research objectives. Section VI addresses the experimental results with a detailed discussion. Finally, section VII concludes this research.

II. IMPORTANCE OF CLASSIFICATION IN NATURAL LANGUAGE

The use of Natural Language Processing (NLP) is increasing and demanding in many fields. Text Classification plays a crucial role in tackling many applications. It is frequently used in technology such as spam detection, virtual assistants, and chatbots. Information generation and sharing requires Text Classification to provide relevant content to the community. Education and Information sciences involve document categorization in Libraries and educational systems. For example, Hadith categorization handles the talks of the Prophet Mohammad (Peace be upon him, PBUH) [2]. Healthcare encompasses the use of text classification in handling a huge amount of information in medicine.

III. RELATED WORKS

This section includes two main parts. The recent works tackling Arabic text Classification based DL are introduced. Then, the related works that applied GA with CNN are discussed.

A. DEEP LEARNING BASED ARABIC TEXT CLASSIFICATION

In [14], the authors aimed at classifying Arabic text using four models Convolutional Neural Networks (CNN), Deep Belief Networks (DBN), Deep Auto Encoders (DAE), and Recursive Auto Encoder (RAE). For vector representation, they applied the Arabic sentiment lexicon (ArSenL) that consists of three polarity levels, Positive, Negative, and Neutral. As for the text pre-processing, the authors did not remove any stop words or applied any text stemming. The Linguistic Data Consortium Arabic Tree Bank (LDC ATB) dataset composed of 1,180 sentences was utilized. Each sentence was represented as a list of words. The dataset was split into 944 sentences for training and 236 sentences for testing. The applied models resulted in 55.5% accuracy rate for CNN, 57.5% for DBN, 60.4% for DAE, and 74.3% for RAE. This low accuracy might be caused due to the absence of the preprocessing phase.

Another research in Arabic text categorization is conducted by [19]. The authors proposed using Markov clustering unsupervised learning and Deep Belief Networks (DBN). Three phases were followed to conduct their experiments. The first phase was the pre-processing that consisted of removing punctuating marks, conjunctions, and using the root form of each word. The next phase was the clustering phase, they applied fuzzy C-means and Markov clustering. Finally, in the last phase, they trained the proposed model using DBN. The proposed method was tested using two datasets, Al-Jazeera consisting of 10,000 instances, and Saudi Press Agency dataset consisting of 6,000 instances. The results achieved a precision of 91.2% and recall of 90.9% giving a 91.02% F1 measure.

The authors in [20] aimed to perform Arabic Sentiment Analysis using Recursive Neural Tensor Networks (RNTN). In their work, they created the first Arabic Sentiment Treebank (ARSENTB). In addition, they used the corpus of QALB that contains 550,273 comments on Al-Jazeera articles. They employed MADAMIRA a morphological analyzer that extracts orthographic and morphological features with high accuracy. Their proposed model improved the results of the other classifiers such as Support Vector Machine (SVM), Recursive Auto-Encoders (RAE) and Long Short-Term Memory (LSTM) by 7.6%, 3.2%, and 1.6% for QALB dataset respectively.

Another research in Arabic text classification [15] conducted on Arabic tweets dataset containing 2,026 reviews

about health services. The data (tweets) was collected using Twitter API, and manually annotated to either positive or negative. Then, the preprocessing was applied to remove the Arabic diacritics, the Tatweel character "-", and the repetitive letters. The authors used Convolutional Neural Networks (CNN) and other Machine Learning techniques (Naïve Bayes, Support Vector Machine, and Logistic Regression) in their experiments. CNN achieved an accuracy rate of 90%, While SVM resulted in an accuracy rate of 91%.

The research done by [21] applied Deep Recurrent Neural Network (RNN) for sentence level Arabic text Classification. They aimed to compare the results of Support Vector Machine (SVM) and RNN. The authors used the Arabic Hotels' reviews dataset that contains 24,028 annotated instances. At first, they extracted the following features: Morphological features, N-grams, syntactic features, and word embedding. The results showed that SVM outperformed RNN with an accuracy rate reaching 93% and 86% respectively. However, RNN was faster than SVM.

The authors in [22] aimed to improve the Arabic text classification by applying Convolutional Neural Networks (CNN). The proposed method consisted of four steps. The authors started with text preprocessing to clean the text from different characters and numbers, remove the stop words, and find the stem of the words. Then, they used Bag of Words method for text representation. After that, they applied Term Frequency-Inverse Document Frequency (TF-IDF) as a dimension reduction method to find the most important words. Finally, they trained the CNN model. The dataset used in this research was collected by the authors from different Moroccan Newspapers consisting of 111,728 documents categorized into five classes. The results achieved an accuracy rate of 92.94%.

The authors in [23] applied Adversarial Deep Averaging Network (ADAN) for Arabic and Chinese sentiment classification. The authors aimed to find a model for cross-lingual sentiment classification. They used the English language that has a huge number of lexical dictionaries to find sentiment classification for other languages (Arabic and Chinese). They applied ADAN with a feedforward network. In their network design, they used three fully connected layers with ReLU activation function, along with Adam optimization, a learning rate of 0.0005 and 30 epochs. Moreover, they implemented the network using PyTorch. To validate the study, the authors used many datasets such as English products reviews, Chinese hotels reviews, labeled and unlabeled Arabic datasets. Also, they used Arabic-English Bilingual Word Embeddings and Chinese-English Bilingual Word Embeddings corpora. The proposed model achieved an accuracy rate of 54.54% for Arabic classification, and 42.49% for Chinese classification.

The authors in [24] proposed a new sentiment analysis classification framework based on a specific structure of CNN for Arabic Dialects. The structure of CNN was called narrow due to its small number of convolutional layers (selected to 3) pursued by the max pooling layer. The model used the row dataset containing tweets without applying lexical features and lexicon resources. Even though the dataset size was small, the proposed model outperformed the existing works and achieved a high recall rate using SemEval-2017 Arabic dialect Twitter datasets.

The authors in [7] proposed a new Arabic text classification framework called A Superior Arabic Text Categorization Deep Model. It is based on a Multi-Kernel CNN and N-gram word embedding. The idea was to enhance the word representation and use the word embedding method in Arabic text. To validate the obtained results, the authors utilized 15 public datasets and performed a comparison study with the well-known Machine and Deep Learning algorithms. The classification accuracy reached a high value of 97.58%, which is superior to the existing results.

In [6], the authors aimed at demonstrating that the stemming techniques can affect the classification accuracy using large Arabic text and Deep Learning. To this end, they employed eleven stemming methods based on stem and root along with seven Deep Learning techniques (CNN, CNN-LSTM (Long Short Memory), Bidirectional LSTM, CNN-GRU (Gated Recurrent Units), Bidirectional GRU, Attention-based GRU, Attention-based LSTM). They utilized the word embedding method. They used two large datasets (Arabic News Texts (ANT v1.1) and Saudi Press agency (SPA) and cross validation with ten folds. The results showed that the stem-based techniques are somewhat better than the root-based stemming techniques. The authors recommended the utilization of the bidirectional and attention learning concepts to deal with Arabic text classification. The F measure achieved 97.96% for Attention-based GRU combining with stem-based method. In addition, they suggested the use of stem based method along with skip-gram representation when handling a small sized vector, but it can be used with CBOW (Bag-Of-Word) when handling a large dimensional vector.

B. CONVOLUTIONAL NEURAL NETWORK AND GENETIC ALGORITHM

Genetic Algorithm is considered to be one of the most reliable optimization algorithms. The following related works discussed its use to optimize the weights of CNN.

In [16], the authors proposed a new approach to recognize human actions. They attempted to optimize the weights of CNN using GA. The weight masks and the seed value were considered as a GA chromosome. The fitness function is represented by the classification accuracy. The CNN was trained using steepest descent algorithm. The authors used chromosomes of size 64 where the first 63 were used to encode the three convolutional masks (A convolutional mask is a small matrix used to create new effects on the image pixels such as blurring), and the last number is for the seed value. Moreover, The convolution masks value range was [−100,100], and the range of seed value was 0 to 5000. The authors applied the crossover and the mutation with a probability 0.8 and 0.01 respectively. After five iterations, the best chromosome was found and used to initialize the

CNN classifier for the testing stage. The proposed method was tested on the action YouTube videos dataset (UCF50) and achieved a result of 96.88%.

Another research [17] applied CNN combined with GA to optimize weight initiation for crack detection in images. The chromosomes were initialized randomly with values ranging between 0 to 255, and Bias values selected between −128 and 128. The chromosome was used as the weights' initial values in the training stage to produce an output image. This output image was compared with the original image to calculate the fitness value of that chromosome. The fittest chromosomes was selected using roulette selection method, with a 1% mutation rate. Furthermore, the crossover point number is randomly selected between 0 and the chromosome length. The proposed approach resulted in a 92.3% accuracy for crack detection of 100 images.

IV. BACKGROUND

This section introduces the three main concepts used in this study including CNN, ANLP, and GA.

A. DEEP LEARNING AND CONVOLUTIONAL NEURAL NETWORK (CNN)

Deep Learning is part of Machine Learning (ML), it uses a more complex architecture of Artificial Neural Networks (ANN). The basic ANN consists of one hidden layer, however in Deep Learning more hidden layers are used. As in ML, DL can be a supervised or unsupervised learning. One of the advantages of using DL over ML is eliminating the need for data pre-processing for the numerical data [25] and feature selection, which makes it more adaptable for new domains and tasks [26].

Convolutional neural network (CNN) is a DL technique. CNN is inspired from the brain function. It is named Convolutional as it performs a mathematical linear operation called ''convolution'' instead of general matrix multiplication [27]. It is known to perform well in processing the data that has a grid-like topology [27]. It can have many dimensions depending on the data that needs to be processed, 1D is for processing signals and text, 2D for processing images or audio, and 3D for video processing.

The basic architecture of CNN includes input layer, an output layer, and a group of hidden layers that consist of several convolutional layers, normalization, pooling, and fully connected layers. The first hidden layer is always a convolutional layer, while the fully connected layer is always the last. The convolutional layer is used to detect the related features from the input data, while the pooling layer merges similar features to one. The fully connected layer converts the input to an N-dimensional vector, where N is the number of classes to be classified [28]. Figure1 illustrates the basic architecture of CNN.

During the learning process, at each iteration, there is a loss function that evaluates the classification quality. It calculates the similarity measure between the network prediction and the existing data. CNN neurons in the first hidden layer will

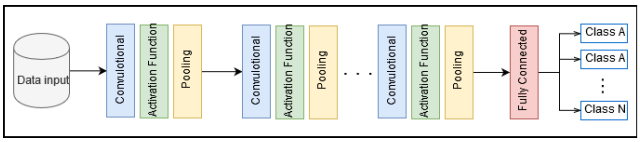


FIGURE 1. Convolutional neural network architecture.

be connected to a small region on the input, this is known as Local Receptive Field (LRF). Each of the LRF has weights and biases that are randomly initialized [29].

In this study, the initialization of the weights is tackled.

After every convolutional layer, there is a non-linear activation function $\alpha()$ applied to the output of the convolutional layer. This is done to enable the CNN to learn non-linear decision boundaries [26]. There are several activation functions that can be used depending on the problem under question such as sigmoid, tanh, ReLU. Changing the activation function can change the results. After the activation function is applied, the data will move to the next layer which is the pooling layer. There are two types of pooling, max pooling and average pooling. Max pooling takes the largest value in each patch of each feature map in a matrix, this type is commonly used [30]. Whereas the average pooling takes the average of that patch. Figure 2 points out this calculation.

Other convolutional layers can be added, however, the last layer is a fully connected layer as illustrated in figure 1.

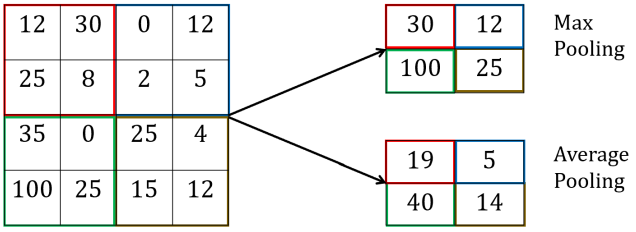


FIGURE 2. The pooling layer calculation.

B. GENETIC ALGORITHM

The Genetic Algorithm (GA) is a search heuristic based on the theory of evolution invented by Charles Darwin [31]. GA was first introduced by John Holland in the early 1970's. The main idea is to follow the process of natural selection and genes as displayed in Figure 3.

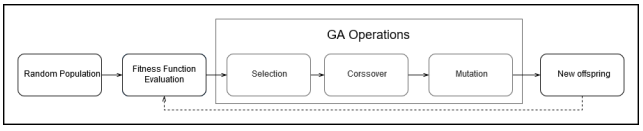


FIGURE 3. Genetic algorithm steps.

Initialize Random Population: the first step consists of generating random solutions. Each solution in the search space is called a Chromosome. This Chromosome involves a set of Genes as illustrated in Figure 4. The number of chromosomes (size of the population) and the chromosome dimension depend on the problem to be solved.

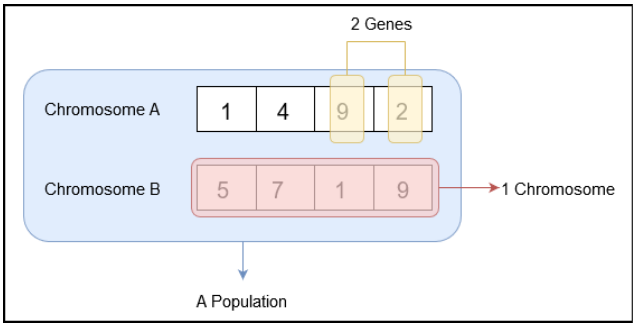


FIGURE 4. Chromosome representation.

Fitness Function: is what the optimization problem is trying to solve [32]. The fitness function evaluates each and every solution in the search space. Therefore, it is one of the most important tasks in GA.

Selection Operator: chooses the good chromosomes from the population to produce a new better population [32]. According to the fitness function evaluation, the fitter the chromosome the most likely it will be selected for the next generation. Different selection methods have been proposed [33], the most well known are the Roulette Wheel and the Linear Rank. The roulette Wheel Selection gives each individual in the population a chance to be selected according to their fitness value. The Linear Rank Selection ranks the individuals depending on their fitness value. The fittest chromosome will have the rank N while the worst chromosome will have the rank 1.

Crossover Operator: is the process of swapping a subset of two chromosomes to create two new chromosomes for the new generation. Usually, the crossover rate is recommended to be high around 70% [34], but it remains a problem-dependent parameter. The three well-known crossover techniques are Single-point crossover, 2-point crossover, and Uniform crossover. An example of a single point crossover is shown in Figure 5.

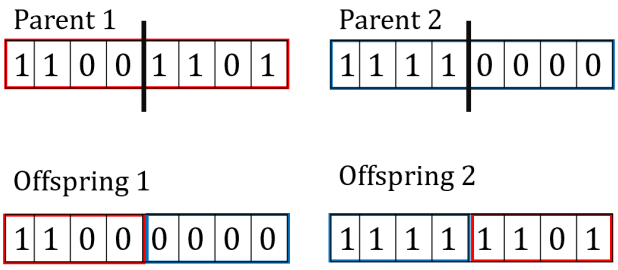


FIGURE 5. A single-point crossover operator.

Mutation Operator: is the process of changing a part of the crossed over chromosome to create a new unique chromosome different from the first generation (the parents). The mutation step helps to avoid sticking in the local optima and finding the global best solution [35]. The mutation happens in a very low probability, ranging between 0.005 to 0.01 [34]. Also, it can happen before the crossover. One type of the mutation is to randomly replace one selected gene. Figure 6 displays the new offspring generated from the previous example (Figure 5), the mutation randomly changed gene 5.

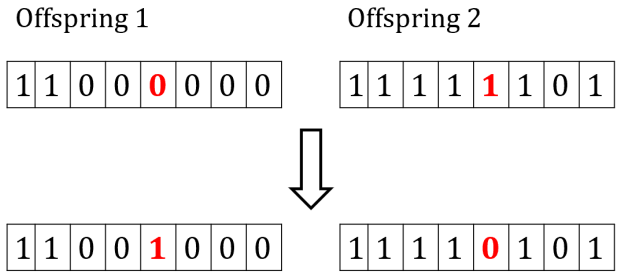


FIGURE 6. The mutation randomly applied to gene 5.

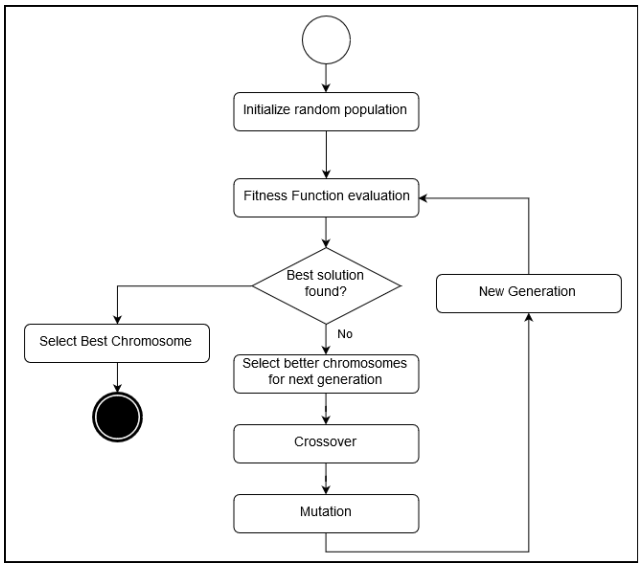


FIGURE 7. Genetic algorithm flowchart.

The three operators will remain on going until a better generation is created, therefore finding the global best solution that did not exist in the first generation. The stopping criteria can be a specific number of iterations. However, the performance of the algorithm is highly dependant on the fitness function used, and the initialization of chromosomes [32]. Other performance factors are related to the population size, mutation and crossover rates, and the total number of iterations performed [36]. GA is used in variety of applications such as ML, image processing, the traveling salesman problem and many more. The GA algorithm flowchart is illustrated in Figure 7

C. ARABIC NATURAL LANGUAGE PROCESSING (ANLP)

Arabic Natural Language Processing is part of NLP that is specified in analyzing and processing Arabic language. Arabic language is complex due its linguistic and derivational structure. There are two types of Arabic, Modern Standard Arabic (MSA) and different regional Arabic dialects, each is written in a different way using different words [37].

Arabic language is different from the English languages in many points. In fact, Arabic is written from right to left. Moreover, in Arabic scripts, there are no capital letters. Also, Arabic letters change their shapes according to their position in the word.

Therefore, NLP tools developed for other languages cannot be used with Arabic language.

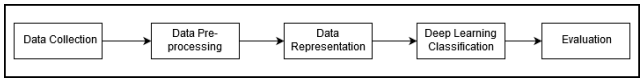


FIGURE 8. ANLP text classification process.

1) ARABIC TEXT CLASSIFICATION

Arabic text classification is a field of ANLP which can be used in many areas for example understanding the user's feedback or classifying different articles and news to pre-defined categories. It is processed through several phases as shown in Figure 8.

1) *Data Collection:* Collecting different texts from different resources.
2) Data Pre-processing: Involves several steps to prepare the data to be processed by the classifier.
   a) *Cleaning:* Removing and correcting missing or incorrect records.
   b) *Normalization:* The process of transforming the text to its basic form.
   c) *Tokenization:* An essential step for any NLP process. It is the process of breaking a sentence into words known as tokens. This process is performed by identifying the words boundaries. For some languages, it is easy to detect the boundaries since it is mostly a white space character [38].
   d) *Stemming:* The process of removing all prefix and suffix from the word, and returning it to the base form (A word root or a word stem).
3) *Data Representation:* Converting the written text to vectors, to feed it to the classifier. In addition, words that have the same meaning will have similar vector representation.
4) *Classification:* The process of assigning a class to a text based on its content. It can be done either through Machine Learning or Deep Learning techniques.
5) *Evaluation:* The results of the classifier is evaluated using the following measures, in order to understand how good the classifier performed:
   a) *Accuracy:* It is the ratio of the correct classified entities to the total number of the dataset.

$$accuracy = \frac{CorrectPredictions}{TotalNumber} \quad (1)$$

   b) *Precision:* It is the ratio of the true positive results to the total number of positives.

$$precision = \frac{TotalTruePositives}{TruePositives + FalsePositives} \quad (2)$$

   c) *Recall:* It is the number of the correct classified entity from one class divided by the total entity number of that class

$$recall = \frac{TotalTruePositives}{TruePositives + FalseNegatives} \quad (3)$$

   d) *F1-Measure:* It is about measuring the test accuracy, the greater the F1 score the better the model is. F1-measure uses both the recall, and the precision scores.

$$F1Score = 2 * \frac{(Recall * Precision)}{(Recall + Precision)} \quad (4)$$

V. RESEARCH METHODOLOGY

The proposed methodology to classify Arabic text is to combine CNN with GA. The steps to be followed are presented in Figure 9.

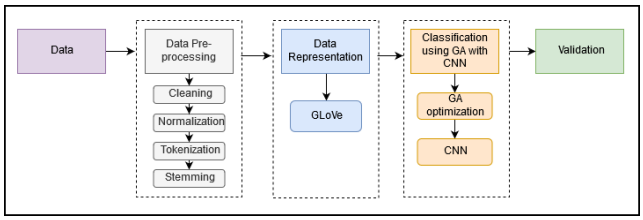


FIGURE 9. GA based CNN proposed methodology.

A. DATA PREPROCESSING AND REPRESENTATION

To make a text ready for classification, several steps of text processing should be performed. As mentioned in section IV-C and Figure 9, the proposed methodology preprocesses the collected documents through cleaning, normalization, tokenization, and stemming.

The cleaning process consists of removing:

- Null values
- Non Arabic characters such as digits, punctuation marks, and extra white spaces.
- Arabic or Hindi numbers

After the cleaning is done, the text normalization is applied to deal with the raw format of the Arabic text which will avoid the noise in training the data. Arabic normalization is performed by:

- Removing "hamza" from the sentence
- Removing diacritics
- Replacing "taa" with "ha" and "shorten alif" with "yaa"
- Removing the tatweel character

The tokenization process is related to dividing the sentence into an array of words known as tokens. The tokenization depends mostly on the white spaces between words to retrieve the tokens.

Stemming is the process of getting the root of the words, usually in Arabic language the roots of most of the words consists of 3 letters. Arabic words usually have several affixes attached to them, stemming removes those affixes and keep the word at its original root

Data representation method used in this study in GLoVe. It is pre-trained words embedding that aims to create word vectors while maintaining the meaning in vector space. Unlike the other word vectors methods, GLoVe is based on co-occurrence matrix, and uses global count statistics instead of only local sentence information. The process starts with creating a co-occurrence matrix. Then, local context is considered using a fixed window size. In this research glove.6B.100d model is used.

B. GA BASED CNN

The CNN classification model will be optimized using GA optimization algorithm to find the best weights.

GA is performed while training the data to find the best model with the best weight values. The best model obtained in the raining phase is used in the testing phase to find the classification accuracy.

1) GENETIC ALGORITHM (GA)

In GA, the chromosomes represent the network weights. The population, composed of a number of chromosomes, is randomly initialized. The number of chromosomes represents the number of weight vectors. The fitness function is the accuracy of the training set. Hence, the optimization problem involves maximizing the accuracy of the training set when applying CNN. Based on the recommendations of [16], the tournament selection of three participants is utilized. The crossover is applied using 2-point with probability of 0.65. The mutation rate is equal to 0.05. GA is iterated many times. The iteration number is set after performing several experiments discussed in section VI. Table 1 presents a summary of the selected parameters.

TABLE 1. Genetic algorithm parameters.

Operation	Value
Fitness Function	Accuracy of CNN
Selection	Tournament
Crossover	0.65
Mutation	0.05

After running the GA and obtaining the weights' optimal values, the CNN is performed to classify the Arabic text using the testing set.

2) CONVULOTIONAL NEURAL NETWORK (CNN)

CNN is used with one dimension since the text data is handled. Different layers are used including the Convolutional layers, the Pooling layers, the Activation functions, and the Fully Connected layer to add the non-linearity to the model. Besides, the Rectified Linear Unit (ReLU) activation function is applied. This choice is based on several research studies ([22], [39], [40]) that proved its efficiency and high accuracy for text classification problems. In addition, max pooling is employed along with a dropout probability of 0.5. Moreover, a maximum gradient of 5.0, a learning rate of 0.0005, and several number of epochs are tested. Since text document is represented in one dimension therefore, 1D Convolutional Layer is used.

The architecture used is as follow:

- Embedding Layer
- 1D Convolutional Layer (Conv1D)
- Max Pooling Layer (MaxPooling1D)
- Relu Activation
- 1D Convolutional Layer (Conv1D)
- Relu Activation
- Max Pooling Layer (MaxPooling1D)
- 1D Convolutional Layer (Conv1D)
- Relu Activation
- Max Pooling Layer (MaxPooling1D)
- Flatten Layer
- Dense Layer

C. VALIDATION

To validate the proposed model, four classification metrics are computed (see IV-C1). The datasets are divided into three sets, 70% for training, 15 % for validation, and 15% for testing. The training data is used to train the model. The validation data is used to select the model based on the best solution (weight vector) achieving the highest accuracy. The testing data is used to evaluate proposed classification model GA-CNN.

VI. EXPERIMENTAL RESULTS

A. DATA COLLECTION

In this article, two datasets are used to experiment the proposed methodology. The first dataset was recently collected [18] from two major news sources in Saudi Arabia, AlRiyadh Newspaper and Saudi Press Agency (SPA). It consists of news text details and news titles classified into six classes: Economical, Political, Social, Sports, General news, and Arts. The dataset has 45,936 records. Table 2 presents the classes and their totals. The dataset can be downloaded [18]

TABLE 2. Details of Saudi newspapers articles dataset.

Resource	Economical	Political	Social	Sports	General news	Arts	Total
AlRiyadh	1,992	3,442	220	3,954	2,362	591	12,561
SPA	5,637	6,126	6,544	3,180	7,786	4,101	33,374
Total	7,629	9,568	6,764	7,134	10,148	4,692	45,935

The second dataset called Moroccan Newspapers Articles Dataset (MNAD) collected by [22] from Moroccan newspapers, Hespres, Akhbarona, and Assabah. It contains 111,728 documents are categorized into five classes as illustrated in Table 3.

TABLE 3. Details of Moroccan newspapers articles dataset.

Resource	Politic	Sports	Economy	Culture	Diverse	Total
Hespress	5,737	6,965	3,795	3,023	7,475	26,995
Akhbarona	12,387	5,313	7,820	5,080	0	30,600
Assabah	2,381	34,244	2,620	5,635	9,253	54,133
Total	20,505	46,522	14,235	13,738	16,728	111,728

B. DATA PREPROCESSING AND REPRESENTATION

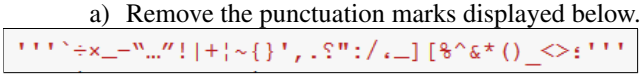
Data preprocessing was implemented using Python 3. As discussed in section V-A, the preprocessing step contains four

main sections: cleaning, normalization, tokenization, and stemming. For both tokenization and stemming, the National Language ToolKit (NLTK) [41] was used. NLTK is an open-source library for NLP that is implemented for different languages including Arabic language. This library provides functions for classification, tokenization, stemming, tagging, and lists of stopwords. The preprocessing steps applied to both datasets are similar, except for SNAD dataset that required extra cleaning to remove some unnecessary text as it was collected from different websites. The text data gathered from the Saudi Press Agency (SPA) required removing the header and the footer. The header includes the date of the news and the location and the footer has a timestamp and the news URL. Figure 10 shows one article before (on the left) and after (on the right) the cleaning. The text in red was removed. Moreover, the news categories (the class attribute) in the dataset was categorical (political, sports, etc..) which was converted to numerical values (0, 1, etc…) to be used with the classifier.

FIGURE 10. An article before (one the right) and after (on the left) cleaning.

After performing extra cleaning to SNAD dataset, the preprocessing steps applied to both datasets are as follows:

1) Loading the dataset in csv file (45,935 Records for SNAD, 111,728 Records for MNAD).
2) *Cleaning:*
   a) Remove the punctuation marks displayed below.



   b) Remove null values.
3) *Normalization:*
   a) Normalize the data using normalize_arabic() method as shown in the appendix and section V-B2.
   b) Remove diacritics using deNoise() method, as shown in the appendix
4) *Tokenization:*
   a) Import NLTK library.
   b) Remove stop words using the stopwords list provided in NLTK library.
   c) Use the word_tokenize() function to get the tokens (shown in the appendix).
5) *Stemming:*
   a) Import the ISRIStemmer Library.
   b) Use stem() function to get the root of each token (see the appendix).

Data representation is done using Global Vectors for Word Representation (GLoVe) [42]. The implementation of GLoVe is within glove_python library. For this research a pre-trained GLoVe model is used called "glove.6B.100d".

C. CLASSIFICATION USING CNN WITH MNAD DATASET

The goal of this experiment is to show the classification results after setting the CNN parameters using MNAD dataset. Note that, the GA is not applied in this experiment, which means that the CNN weights are randomly initialized.

To set the parameters, the training (70% representing 78,209 records) and validation (15% representing 16,759 records) sets are utilized. Running the classifier using different combinations of CNN parameters can change the results significantly. The main parameters used are as follow:

- *Epochs:* Number of iterations.
- *Batch_Size:* Number of samples per gradient update.
- *Loss Function:* A method used to measure the inconsistency between predicted and real values.
- *Optimizer:* A method to adjust the weights according to the input values.
- *Classification Accuracy:* The method to be used to evaluate the results.

For that, many attempts are performed and their results are logged as shown in Table 4. As seen, three different optimizers (Adam, AdaDelta, RMSprop) are used with different number of batch sizes (100, 250, 500, 1000) and epochs (10, 15, 20, 30, 40, 50, 60). Also, for each attempt, the training accuracy (Train) and validation accuracy (Val) are logged along with the run time in hours. As mentioned in section V-B2, Relu activation function is used.

In order to select the right parameters' value achieving a high accuracy, Figures 11, 12 and 13 are drawn based on the results obtained in Table 4.

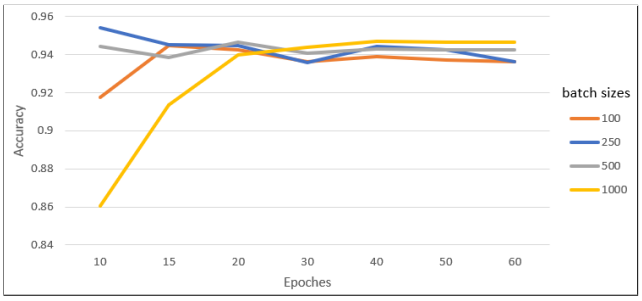


FIGURE 11. Parameters setting of CNN with MNAD dataset using RMSprop optimizer.

In Figure 11, the training accuracy results for RMSprop optimizer are illustrated for different batch sizes over an increasing number of epochs. It can be noticed that the batch sizes of 250, 500 started with a higher accuracy rate, but the accuracy keeps increasing and decreasing for different

TABLE 4. CNN parameters setting using MNAD dataset.

Epochs	Batch	Optimizer	Train	Val	Time
	100	Adadelta	0.9421	0.9194	2:00:10
		Adam	0.9914	0.9414	3:20:24
		rmsprop	0.9749	0.9175	2:10:34
	250	Adadelta	0.9191	0.913	2:06:12
		Adam	0.9876	0.945	2:20:42
		rmsprop	0.9714	0.954	2:35:14
	500	Adadelta	0.8616	0.7484	2:14:02
		Adam	0.9923	0.9403	1:56:23
		rmsprop	0.9538	0.9445	1:57:15
	1000	Adadelta	0.7816	0.7888	1:55:01
		Adam	0.9791	0.9303	2:01:24
		rmsprop	0.9167	0.8606	1:56:45
15	100	Adadelta	0.9707	0.931	4:19:21
		Adam	0.9916	0.9376	3:56:50
		rmsprop	0.9858	0.9447	3:33:50
	250	Adadelta	0.9296	0.9227	2:32:02
		Adam	0.9965	0.9353	2:50:48
		rmsprop	0.9855	0.9454	2:44:37
	500	Adadelta	0.912	0.8964	2:00:34
		Adam	0.9973	0.9439	1:56:34
		rmsprop	0.9768	0.9384	2:15:45
	1000	Adadelta	0.8202	0.8018	1:59:31
		Adam	0.991	0.9437	2:00:01
		rmsprop	0.943	0.9137	2:02:35
20	100	Adadelta	0.9863	0.9243	5:46:08
		Adam	0.9958	0.9412	5:15:16
		rmsprop	0.9916	0.9427	4:46:01
	250	Adadelta	0.9575	0.9098	3:59:26
		Adam	0.997	0.9428	3:48:14
		rmsprop	0.9908	0.9448	3:35:59
	500	Adadelta	0.9133	0.8971	2:40:40
		Adam	0.9972	0.9413	3:18:51
		rmsprop	0.9873	0.9465	2:25:21
	1000	Adadelta	0.8494	0.8599	2:20:00
		Adam	0.9964	0.9411	2:20:12
		rmsprop	0.9631	0.94	2:24:17
30	100	Adadelta	0.9938	0.9156	8:57:16
		Adam	0.996	0.9449	8:02:58
		rmsprop	0.9946	0.9364	7:18:05
	250	Adadelta	0.9877	0.9372	6:07:02
		Adam	0.9979	0.9464	5:48:04
		rmsprop	0.994	0.936	3:48:09
	500	Adadelta	0.953	0.9279	5:15:04
		Adam	0.9978	0.9388	5:07:38
		rmsprop	0.9927	0.9406	4:56:21
	1000	Adadelta	0.9096	0.8883	3:33:02
		Adam	0.9971	0.939	4:45:01
		rmsprop	0.9842	0.944	6:34:59
40	100	Adadelta	0.996	0.9269	11:38:17
		Adam	0.9976	0.9502	10:34:29
		rmsprop	0.9954	0.939	9:28:27
	250	Adadelta	0.9948	0.9293	8:08:12
		Adam	0.9979	0.9428	7:39:26
		rmsprop	0.996	0.9445	5:24:02
	500	Adadelta	0.9721	0.9	6:54:04
		Adam	0.9907	0.941	6:41:37
		rmsprop	0.9945	0.9428	6:31:03
	1000	Adadelta	0.9286	0.9141	6:25:09
		Adam	0.99	0.935	6:19:00
		rmsprop	0.9886	0.9469	6:07:59
50	100	Adadelta	0.9959	0.9343	14:34:08
		Adam	0.997	0.9411	13:12:30
		rmsprop	0.9962	0.9372	11:58:28
	250	Adadelta	0.9933	0.9339	10:03:34
		Adam	0.998	0.9448	9:30:39
		rmsprop	0.9962	0.9424	9:01:04
	500	Adadelta	0.9897	0.9201	8:33:41
		Adam	0.9979	0.9469	8:17:57
		rmsprop	0.9957	0.9424	8:06:27
	1000	Adadelta	0.9559	0.9065	7:50:59
		Adam	0.92	0.9	7:43:28
		rmsprop	0.9902	0.9467	7:39:45
60	100	Adadelta	0.9961	0.9257	17:28:51
		Adam	0.9974	0.9427	15:52:46
		rmsprop	0.996	0.9365	14:18:25
	250	Adadelta	0.9897	0.9202	14:19:30
		Adam	0.9981	0.949	11:22:50
		rmsprop	0.997	0.9365	10:45:14
	500	Adadelta	0.9929	0.9289	12:35:33
		Adam	0.9865	0.9	9:57:15
		rmsprop	0.996	0.9425	9:39:59
	1000	Adadelta	0.9426	0.923	11:14:59
		Adam	0.9978	0.9452	9:16:29
		rmsprop	0.9967	0.9467	9:08:13

epochs. However, the batch size of 100 started with a low accuracy and increased by increasing the epochs, but it does not show stability in the results. On the other hand, the batch size of 1000 started with the lowest accuracy of 86% but it increased stably by increasing the epochs. Also, it can be seen that the results are stable when reaching ephoces 40.

However, the training accuracy results for Adam and Adadelta optimizers showed instability for different batch sizes over the tested epochs as shown in Figures 12 and 13. The results for both have an irregular pattern, but more stable results can be achieved by increasing the number of epochs which can increase the running time. Consequently, the RMSprop optimizer is selected as the best choice for this dataset. To sum up, the best parameters for MNAD dataset are:

- Epochs: 40
- Batch size: 1000
- Optimizer: RMSprop
- Training Accuracy: 98.86%
- Validation Accuracy: 94.69%
- Run Time: 6:07:59 hours

After training the classifier and finding the best parameters, the obtained model is tested using the test set (15% = 16,759 records). The accuracy reached 94.1%

D. CLASSIFICATION USING CNN WITH SNAD DATASET

Alike the previous experiment in VI-C, CNN is used to find the classification accuracy using SNAD dataset. GA is not yet

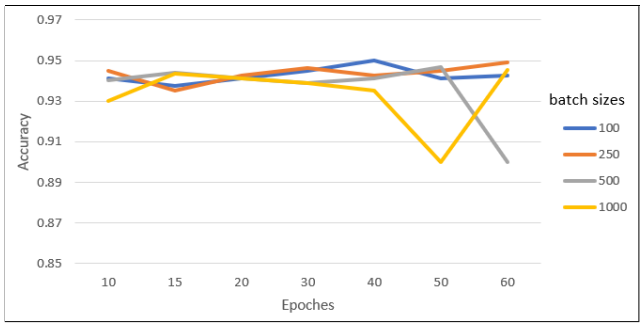


FIGURE 12. Parameters setting of CNN with MNAD dataset using Adam optimizer.

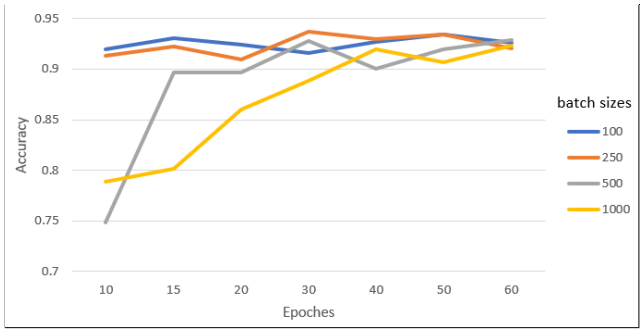


FIGURE 13. Parameters setting of CNN with MNAD dataset using Adadelta optimizer.

employed. The parameter setting is performed using the same batch sizes (100, 250, 500, 1000) and epochs (10,15, 20, 30, 40, 50, 60) values as previously.

Table 5 displays the training (train) and validation (val) accuracies along with the run time spent in hours For each training attempt.

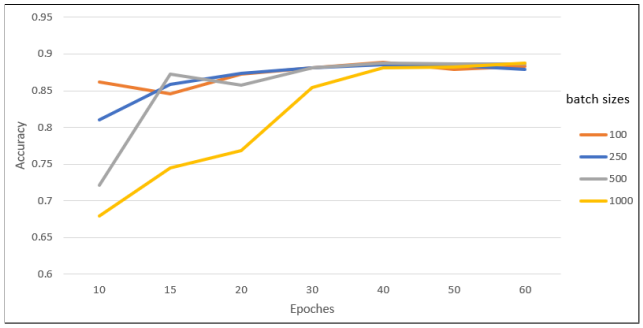


FIGURE 14. Parameters setting of CNN with SNAD dataset using RMSprop optimizer.

Overall, RMSprop optimizer (see Figure 14) started with low accuracy for different batch sizes. The results of batch size 100, has the most unstable results therefore, it is excluded. The results of batch size 250, increased stably but it started decreasing at 50 epochs. However, the results of batch sizes 500 and 1000 are interesting. In contrast, the results of using Adam optimizer (see Figure 15) and Adadelta optimizer (see Figure 16) showed instability using different batch sizes. The obtained results are increasing and decreasing over different number of epochs. Hence, this

TABLE 5. CNN parameters setting using SNAD datase.

Epochs	Batch	Optimizer	Train	Val	Time
10	100	Rmsprop	0.9262	0.8617	0:29:19
		Adam	0.9752	0.8748	0:31:22
		Adadelta	0.8511	0.7695	0:47:52
	250	Rmsprop	0.8375	0.81	0:34:16
		Adam	0.9727	0.861	0:26:12
		Adadelta	0.7614	0.7086	0:40:32
	500	Rmsprop	0.7798	0.7213	0:28:42
		Adam	0.9401	0.8134	0:25:55
		Adadelta	0.5564	0.5475	0:38:06
	1000	Rmsprop	0.6181	0.6796	0:32:32
		Adam	0.9083	0.8366	0:34:08
		Adadelta	0.4383	0.450	0:37:45
15	100	Rmsprop	0.9616	0.8459	0:51:05
		Adam	0.9812	0.8548	0:55:31
		Adadelta	0.9287	0.8467	1:11:38
	250	Rmsprop	0.9292	0.8585	0:55:58
		Adam	0.9858	0.8842	0:46:18
		Adadelta	0.8348	0.7731	01:00:51
	500	Rmsprop	0.8721	0.8125	0:41:57
		Adam	0.9515	0.8823	0:51:28
		Adadelta	0.6872	0.6701	0:57:12
	1000	Rmsprop	0.7711	0.7445	0:56:14
		Adam	0.887	0.8565	0:38:01
		Adadelta	0.5206	0.5241	0:55:58
20	100	Rmsprop	0.9781	0.873	1:25:34
		Adam	0.9894	0.875	1:45:01
		Adadelta	0.9580	0.8443	1:34:48
	250	Rmsprop	0.9569	0.8735	0:48:05
		Adam	0.9919	0.8904	1:14:05
		Adadelta	0.9073	0.8004	1:20:44
	500	Rmsprop	0.9137	0.8577	0:46:55
		Adam	0.9869	0.877	1:09:14
		Adadelta	0.7548	0.6792	1:16:19
	1000	Rmsprop	0.8467	0.7689	1:13:52
		Adam	0.9789	0.8789	0:46:03
		Adadelta	0.5902	0.5303	1:14:23
30	100	Rmsprop	0.9852	0.8812	1:51:17
		Adam	0.9902	0.8651	1:36:43
		Adadelta	0.9786	0.8417	2:23:57
	250	Rmsprop	0.9809	0.8809	1:12:35
		Adam	0.9913	0.8813	1:58:39
		Adadelta	0.9483	0.8634	2:00:43
	500	Rmsprop	0.9601	0.8809	2:23:41
		Adam	0.9917	0.886	1:21:59
		Adadelta	0.8528	0.7918	1:54:21
		Rmsprop	0.9139	0.8542	2:29:08
	1000	Adam	0.9908	0.8925	1:51:08
		Adadelta	0.6942	0.6340	1:52:20
40	100	Rmsprop	0.988	0.8883	2:53:24
		Adam	0.9906	0.8813	2:44:16
		Adadelta	0.9871	0.8619	3:09:27
	250	Rmsprop	0.9887	0.8853	2:34:43
		Adam	0.9934	0.8816	4:10:11
		Adadelta	0.9748	0.8638	2:41:08

optimizer is not selected. To sum up, the best parameters for SNAD dataset are:

- Epochs: 40
- Batch size: 1000

TABLE 5. *(Continued.)* CNN parameters setting using SNAD datase.

	500	Rmsprop	0.9757	0.8879	1:31:18
		Adam	0.9941	0.8852	2:31:29
		Adadelta	0.9168	0.8462	2:32:55
	1000	Rmsprop	0.9431	0.8808	2:26:52
		Adam	0.991	0.884	2:29:29
		Adadelta	0.7590	0.7369	2:29:06
50	100	Rmsprop	0.9878	0.8791	3:37:16
		Adam	0.9923	0.8793	4:47:08
		Adadelta	0.9896	0.7679	3:57:45
	250	Rmsprop	0.9889	0.8843	3:13:24
		Adam	0.9925	0.8803	3:18:48
		Adadelta	0.9792	0.8412	3:21:48
	500	Rmsprop	0.9816	0.8869	3:06:20
		Adam	0.9838	0.8704	3:08:19
		Adadelta	0.8322	0.8008	3:06:14
	1000	Rmsprop	0.9624	0.8827	3:03:53
		Adam	0.993	0.8692	3:05:34
		Adadelta	0.8670	0.6760	3:06:04
60	100	Rmsprop	0.9894	0.8829	4:49:50
		Adam	0.9921	0.881	4:56:09
		Adadelta	0.9920	0.8623	4:44:13
	250	Rmsprop	0.9916	0.8788	4:51:54
		Adam	0.9936	0.8877	5:02:14
		Adadelta	0.9827	0.8008	4:03:10
	500	Rmsprop	0.9882	0.8869	4:55:43
		Adam	0.99	0.8943	5:02:43
		Adadelta	0.9078	0.8589	3:49:26
	1000	Rmsprop	0.9686	0.8875	4:57:02
		Adam	0.9945	0.8814	5:22:07
		Adadelta	0.9083	0.8128	3:44:08

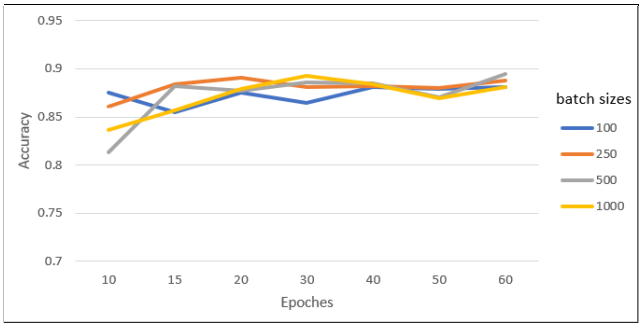


FIGURE 15. Parameters setting of CNN with SNAD dataset using Adam optimizer.

- Optimizer: RMSprop
- Training Accuracy: 94.31%
- Validation Accuracy: 88.08%
- Run Time: 2:26:52 hours

After training the classifier and finding the best parameters, the obtained model is tested using the test set. The accuracy reached 84.32%

E. CLASSIFICATION USING GA-CNN

In this experiment, GA is applied to solve the issue of the randomly initialized weights in CNN. The GA operators are introduced in section V-B1 and set to Tournament Selection, a two-points Crossover with a probability 0.65, and a Mutation with a rate 0.05.

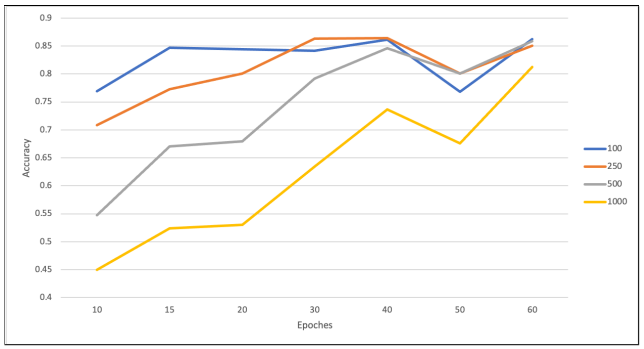


FIGURE 16. Parameters setting of CNN with SNAD dataset using Adadelta optimizer.

After setting the parameters of CNN (Epochs: 40, Batch Size: 1000, Optimizer: RMSprop) for both datasets, the classifier is trained using GA.

In fact, GA randomly initializes the chromosomes, which represent the weights. After running the first iteration and finding the best chromosome (solution), it is used to train the classification model and find the accuracy using the validation set. This process is iterated 100 times. Afterwards, the best accuracy is selected along with the best chromosome. The best values of the weights (the best chromosome) are then used to find the classification accuracy for the testing set.

It is worth noting that the number of iterations is set after performing several trials to insure that the obtained results cannot be enhanced anymore.

1) CLASSIFICATION USING GA-CNN FOR SNAD DATASET

Table 6 figures out the accuracy results for the baseline (CNN without GA) and GA-CNN using SNAD datasets for training/validation and testing sets. It is clear that GA-CNN improved the classification accuracy.

TABLE 6. Classification accuracy for the baseline and GA-CNN using SNAD dataset.

	CNN	GA-CNN
Validation	0.8808	0.9423
Testing	0.8432	0.8871

Additionally, Table 7 provides the results summary using the four classification metrics along with the Root Mean Square Error (RMSE) before and after integrating GA to CNN. GA-CNN outperformed the baseline. This improvement reached 4.39% in terms of accuracy. The RMSE obtained by CNN is 0.0317 greater than that obtained by GA-CNN with a value equals to 0.0182. The RMSE was reduced by 0.0135 when applying GA-CNN.

Unfortunately, it is not possible to tackle a comparison study since this dataset is new and not yet used.

2) CLASSIFICATION USING GA-CNN FOR MNAD DATASET

The classification accuracy and the RMSE attained 98.42% and 0.0153 respectively for the testing set when applying GA-CNN. It is shown above that the accuracy (resp. RMSE)

TABLE 7. Classification results for the baseline and GA-CNN using SNAD testing set.

Measure	Accuracy	F1 Score	Precision	Recall	RMSE
CNN	0.8432	0.8584	0.8704	0.8499	0.0317
GA-CNN	0.8871	0.8920	0.8970	0.8871	0.0182

achieved 94.10% (resp. 0.0259) for the same testing set when applying the baseline. This result proved again the improvement that GA-CNN produces on Arabic text Classification.

MNAD dataset was already used in [22]. Then, a comparison result is performed. Table 8 presents a comparison between these two studies (the proposed study and [22]). As indicated, both studies used the same preprocessing steps but different representation methods. The proposed GA-CNN and the baseline outperformed the existing study with 5.48% and 1.16% respectively in terms of accuracy. The RMSE was not mentioned in [22].

TABLE 8. MNAD dataset - results of the comparison study.

	[22]	Proposed Research	
		CNN	GA-CNN
Data Size	111,728	111,728	
Pre-porcessing	- Remove Stop Words - Stemming - Remove Punctuation - Remove Digits	- Remove Stop Words - Stemming - Remove Punctuation - Remove Digits	
Representation	TF-IDF	Glove	
Accuracy	0.9294	0.9410	0.9842
RMSE	-	0.0259	0.0153

F. DISCUSSION

Many experiments were carried out in this article to accomplish the four research objectives. Firstly, the existing works were deeply investigated to bring out their limitations. It was stressed that CNN was not combined to any optimization method to classify Arabic text. Consequently, the first objective was reached. Besides, since Arabic language is complex and requires a huge effort to prepare the documents for processing, an extensive preprocessing phase was implemented on both datasets (SNAD and MNAD). This phase involves the main operations including cleaning, normalization, tokenization, and stemming. In addition, a recent data representation method (Glove) was applied to convert text documents to numeric vectors. This preprocessing phase and mainly the employment of Glove enhanced the classification accuracy as shown in Table 8. Hence, the second objective is effectively met.

To demonstrate the efficiency of the proposed model GA-CNN, two datasets were utilized and four experiments were performed. The first and second experiments consist of classifying both datasets using CNN. Each of these experiments started with setting the parameters of CNN including the epochs, the batch size, and the optimizer. Both datasets were divided into training, validation and testing. The training and validation sets were used. After many trials, the best model with the highest classification accuracy was obtained setting the above parameters to 40, 1000, and RMSprop respectively for both datasets. The training and validation accuracy values reached 98.86% and 94.69% respectively for MNAD dataset. While they achieved 94.31% and 88.08% respectively for SNAD dataset.

Figure 17 shows the best training and validation accuracy values for both datasets using the same parameters set above. The training and validation accuracy rates for both datasets were satisfactory, even though the accuracy of MNAD is higher than that of SNAD. It is clear that MNAD dataset performed better than SNAD dataset. This later requires further investigation especially in preprocessing. Moreover, there is no significant difference between the accuracy of the training and validation for both datasets, this difference reached 4.17% for MNAD and 6.23% for SNAD. The validation accuracy is less than training accuracy because the model is already familiar with the training data contrary to the validation data which is a new dataset. So, the results indicates that the model of the training data qualifies to the validation data.

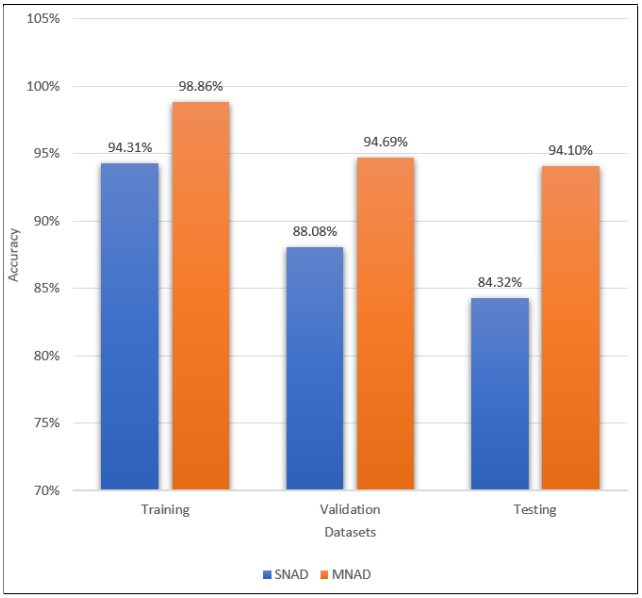


FIGURE 17. Accuracy of the training, validation, and testing for both datasets using the parameters set above.

This results can be distinctly shown in Figures 18 and 19 where the accuracy curves of the training and validation are represented throughout 60 epochs using 1000 batch size and RMSprop optimizer for both datasets respectively. Thus, Both training and validation curves are convergent for both datasets.

The run-time attained 6 hours for MNAD and 2 hours 26 min for SNAD. For the run-time, MNAD took long time due to its size reaching 111,728 documents contrary to SNAD which contains only 45,955 documents.

The last experiments involved the classification of MNAD and SNAD using GA-CNN. Recall that, the same parameters' values (for the batch sizes, the optimizer, and the epochs), obtained above, were used in these experiments. For SNAD dataset, no comparison with the state-of-the art could

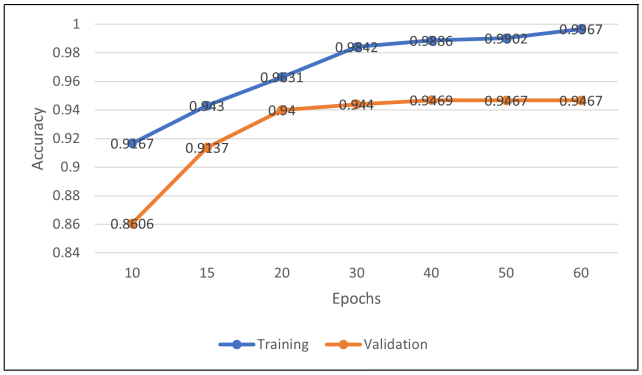


FIGURE 18. The training and validation accuracy curve for MNAD dataset using batch size equals 1000 and RMSprop optimizer.

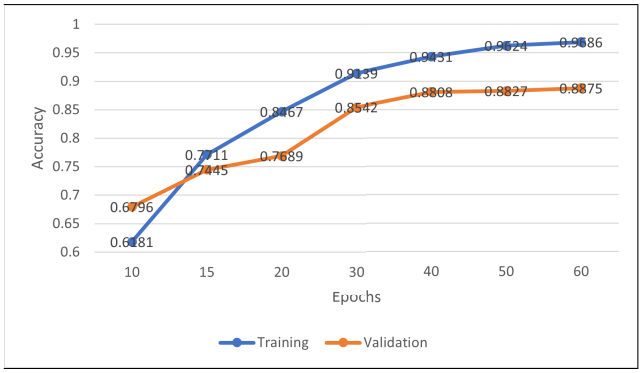


FIGURE 19. The training and validation accuracy curve for SNAD dataset using batch size equals 1000 and RMSprop optimizer.

be realized as it is a new dataset and the first time used. However, a comparison was performed between the baseline and GA-CNN. Table 6 presents the accuracy of the validation and testing sets using the baseline and GA-CNN. It is clearly indicated that GA-CNN outperformed the baseline in both classification stages (validation and testing). GA-CNN enhanced the accuracy and the RMSE in the testing stage by 4.39% and 0.0135 respectively. Table 7 shows the five classification metrics using the testing set for both models. Again, GA-CNN is the best model.

The last experiment figures out a comparison study between the baseline, GA-CNN, and one related work using MNAD. As indicated in Table 8, both the baseline and GA-CNN outperformed the related work [22], and GA-CNN achieved the first place. In fact, the baseline (CNN alone) is superior than [22] even though the same dataset (MNAD) was used in both studies with the same size and preprocessing operations. The only explanation that can justify this achievement is the use of Glove in representing the data (instead of TF-IDF). GA-CNN reached an achievement of 5.48%. Unfortunately, the running time could not be compared because it is not mentioned in [22].

Furthermore, Figure 17 shows the best accuracy of the training, validation and testing for both datasets. As expected, MNAD dataset achieved a better accuracy than SNAD dataset in the testing stage. Again, the validation accuracy is greater than the testing accuracy which is normal because the model's hyperparameters have been adjusted precisely for the validation dataset. The difference between the validation and testing accuracy is not important, resulted in 0.59% for MNAD and 3.76% for SNAD dataset.

Consequently, this study proposed a new hybrid and competitive classification model for Arabic text classification. The results of the experiments insure the accomplishment of objectives 3 and 4.

1) RESEARCH IMPACTS

The research impacts of the proposed hybrid model are essential and emergent.

- Supply a new Arabic Text Classification model that fill in the gap in ANLP.
- Provide a hybrid classifier based on GA-CNN that enhances the classification accuracy by an average of 4 - 5%.
- Contribute in enhancing a Deep Learning technique by integrating an optimization algorithm to find the best weights.

2) ADVANTAGES AND LIMITATIONS OF GA-CNN

The present research has three main advantages.

- GA-CNN is a competitive and efficient classification model for Arabic text.
- The parameter setting of CNN are deeply investigated and hence can be used in future study when handling the same datasets
- The new dataset SNAD is used for the first time. This allows new and future comparison studies.

The limitation of the proposed model is that it takes time when training data using GA to find the optimal weights.

VII. CONCLUSION

This study proposed a hybrid classification model for Arabic text based on CNN and GA. The proposed model was validated using two large datasets. GA-CNN yielded excellent results. Moreover, this model was successfully compared with the baseline and an existing method. Therefore, the combination of CNN with GA to improve the classification accuracy and the RMSE for Arabic text was validated. For the computation time, GA-CNN takes more than the baseline due the execution of GA in the training/validation phase to produce the best weights. Based on this study, some future research could be suggested:

- Investigate the parameter setting of GA to find the best values that can enhance the hybridization CNN-GA.
- Combine another optimization method such that Particle Swarm Optimization with CNN to enhance more the classification accuracy in Arabic text.
- Use large datasets to test the extent of GA-CNN in enhancing the accuracy of Arabic text classification.
- Explore other Deep Learning techniques to classify Arabic text.

APPENDIX

```
#Text pre-processing Python code
#Load the data
arabic_data =
pd.read_csv('data/arabic_dataset.csv');

#Normalization code
text = arabic_data['text']
def normalize_arabic(text):
    text = re.sub("[]", "", str(text))
    text = re.sub("", "", str(text))
    text = re.sub("", "", str(text))
    text = re.sub("", "", str(text))
    text = re.sub("", "", str(text))
    text = re.sub("", "", str(text))
    return text

#Remove vowels
def deNoise(text):
    noise = re.compile("""    | # Tashdid
                          | # Fatha
                          | # Tanwin Fath
                          | # Damma
                          | # Tanwin Damm
                          | # Kasra
                          | # Tanwin Kasr
                          | # Sukun
                          # Tatwil/Kashida
                      """, re.VERBOSE)
    text = re.sub(noise, '',  str(text))
    return text

#Remove punctuations
arabic_punctuations = '''`÷×<>_()*&^/:
".,'{}~¦+|!”...“-'''
english_punctuations =
 ↪  string.punctuation
punctuations_list = arabic_punctuations
+ english_punctuations

def remove_punctuation(text):
    translator = str.maketrans('', '',
    punctuations_list)
    return text.translate(translator)

remove_punctuation(
    str(arabic_data['text']))

#Remove nulls
arabic_data.dropna(axis=0, how='any',
thresh=None, subset=None, inplace=True)

# Remove the Stop words and stemming
import nltk
from nltk.corpus import stopwords
from nltk.stem.isri import ISRIStemmer

def fun(i):
    return macro_to_id[i]
print('[+] Removing the stop words')
nltk.download('punkt')
nltk.download('stopwords')
stop_words=set(stopwords.words('arabic'))
st = ISRIStemmer()
macronum=sorted(set(dataset['targe']))
macro_to_id = dict((note, number) for
 ↪  number, note in enumerate(macronum))
dataset['targe']=dataset['targe'].apply(fun)
texts = []
labels = []
for idx in range(dataset.text.shape[0]):
    text =
     ↪  BeautifulSoup(dataset.text[idx])
    texts.append(clearnFunc(str
        (text.get_text().encode())))
for idx in dataset['targe']:
    labels.append(idx)
def filter_stop_words(texts,
 ↪  stop_words):
    for i, sentence in enumerate(texts):
        new_sent = [word for word in
         ↪  sentence.split() if word not
         ↪  in stop_words]
        texts[i] = ' '.join(new_sent)
    return texts
stop_words =
 ↪  set(stopwords.words("arabic"))
texts = filter_stop_words(texts,
 ↪  stop_words)

# Tokenization
print ('[+] Tokenization')
tokenizer =
 ↪  Tokenizer(num_words=NUMBER_OF_WORDS)
tokenizer.fit_on_texts(texts)
sequences =
 ↪  tokenizer.texts_to_sequences(texts)
word_index = tokenizer.word_index
print('[+] The number of unique Tokens
 ↪  :',len(word_index))
data = pad_sequences(sequences,
 ↪  maxlen=SEQ_LEN)
labels =
 ↪  to_categorical(np.asarray(labels))
```

```
#CNN Model
from array import array
from keras import backend as K
import tensorflow as tf
import gc

batch_sizes = [1000, 250 ,500,100]
epochs_values = [10,15,20,30,40,50,60]
optimizers_values = ['rmsprop' , 'adam'
 ↪  'Adadelta']

def create_model():
    sequence_input =
     ↪  Input(shape=(SEQ_LEN,),
     ↪  dtype='int32')
    embedded_sequences =
     ↪  embedding_layer(sequence_input)
    l_cov1= Conv1D(128, 5,
     ↪  activation='relu')(embedded_sequences)
```

```
    l_pool1 = MaxPooling1D(5)(l_cov1)
    l_cov2 = Conv1D(128, 5,
     ↪  activation='relu')(l_pool1)
    l_pool2 = MaxPooling1D(5)(l_cov2)
    l_cov3 = Conv1D(128, 5,
     ↪  activation='relu')(l_pool2)
    l_pool3 = MaxPooling1D(35)(l_cov3)
     ↪  # global max pooling
    l_flat = Flatten()(l_pool3)
    l_dense = Dense(128,
     ↪  activation='relu')(l_flat)
    preds = Dense(len(macronum),
     ↪  activation='softmax')(l_dense)
    model = Model(sequence_input, preds)
    return model

#Train CNN
for i in epochs_values:
  for x in batch_sizes:
    for y in optimizers_values:
       test =create_model()
       test.compile(
       loss='categorical_crossentropy',
              optimizer= y,
              metrics=['acc'])
       test.summary()
       print("---------------------")
       print("Epoches:  {0} - BatchSize:
        ↪  {1} - optimizer:
        ↪  {2}".format(i, x, y))
       print("---------------------")
       history=test.fit(x_train,
        ↪  y_train,
        ↪  validation_data=(x_val,
        ↪  y_val),epochs=i,
        ↪  batch_size=x)
       print(">>>END<<<")
```

ACKNOWLEDGMENT

The authors would like to acknowledge the support of Prince Sultan University for paying the Article Processing Charges (APC) of this publication. This work was also supported by the Emerging Intelligent Autonomous Systems Data Science and Blockchain Lab (EIAS), Prince Sultan University, Riyadh, Saudi Arabia.

REFERENCES

[1] B. Comrie, *The World's Major Languages*. London, U.K.: Routledge, 2009.

[2] S. L. Marie-Sainte, N. Alalyani, S. Alotaibi, S. Ghouzali, and I. Abunadi, "Arabic natural language processing and machine learning-based systems," *IEEE Access*, vol. 7, pp. 7011–7020, 2019.

[3] M. Eltay, A. Zidouri, and I. Ahmad, "Exploring deep learning approaches to recognize handwritten arabic texts," *IEEE Access*, vol. 8, pp. 89882–89898, 2020.

[4] M. Al-Smadi, S. Al-Zboon, Y. Jararweh, and P. Juola, "Transfer learning for arabic named entity recognition with deep neural networks," *IEEE Access*, vol. 8, pp. 37736–37745, 2020.

[5] M. T. B. Othman, M. A. Al-Hagery, and Y. M. E. Hashemi, "Arabic text processing model: Verbs roots and conjugation automation," *IEEE Access*, vol. 8, pp. 103913–103923, 2020.

[6] H. A. Almuzaini and A. M. Azmi, "Impact of stemming and word embedding on deep learning-based arabic text categorization," *IEEE Access*, vol. 8, pp. 127913–127928, 2020.

[7] M. Alhawarat and A. O. Aseeri, "A superior arabic text categorization deep model (SATCDM)," *IEEE Access*, vol. 8, pp. 24653–24661, 2020.

[8] S. L. Marie-Sainte and N. Alalyani, "Firefly algorithm based feature selection for arabic text classification," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 32, no. 3, pp. 320–328, Mar. 2020.

[9] D. Khurana, A. Koli, K. Khatter, and S. Singh, "Natural language processing: State of the art, current trends and challenges," 2017, *arXiv:1708.05148*. [Online]. Available: https://arxiv.org/abs/1708.05148

[10] T. Sadad, A. Rehman, A. Munir, T. Saba, U. Tariq, N. Ayesha, and R. Abbasi, "Brain tumor detection and multi-classification using advanced deep learning techniques," *Microsc. Res. Technique*, vol. 84, no. 6, pp. 1296–1308, 2021.

[11] M. A. El-Affendi, K. Alrajhi, and A. Hussain, "A novel deep learning-based multilevel parallel attention neural (MPAN) model for multidomain arabic sentiment analysis," *IEEE Access*, vol. 9, pp. 7508–7518, 2021.

[12] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[13] W. Yin, K. Kann, M. Yu, and H. Schütze, "Comparative study of CNN and RNN for natural language processing," 2017, *arXiv:1702.01923*. [Online]. Available: https://arxiv.org/abs/1702.01923

[14] A. A. Sallab, H. Hajj, G. Badaro, R. Baly, W. El Hajj, and K. B. Shaban, "Deep learning models for sentiment analysis in arabic," in *Proc. 2nd Workshop Arabic Natural Lang. Process.*, 2015, pp. 9–17.

[15] A. M. Alayba, V. Palade, M. England, and R. Iqbal, "Arabic language sentiment analysis on health services," in *Proc. 1st Int. Workshop Arabic Script Anal. Recognit. (ASAR)*, Apr. 2017, pp. 114–118.

[16] E. P. Ijjina and K. M. Chalavadi, "Human action recognition using genetic algorithms and convolutional neural networks," *Pattern Recognit.*, vol. 59, pp. 199–212, Nov. 2016.

[17] R. Oullette, M. Browne, and K. Hirasawa, "Genetic algorithm optimization of a convolutional neural network for autonomous crack detection," in *Proc. Congr. Evol. Comput.*, vol. 1, Jun. 2004, pp. 516–521.

[18] D. AlSaleh, M. BinAlAmir, and S. Larabi-Marie-Sainte, "SNAD arabic dataset for deep learning," in *Intelligent Systems and Applications* (Advances in Intelligent Systems and Computing), K. Arai, S. Kapoor, and R. Bhatia, Eds., vol. 1250. Cham, Switzerland: Springer, 2021.

[19] V. Jindal, "A personalized Markov clustering and deep learning approach for arabic text categorization," in *Proc. ACL Student Res. Workshop*, 2016, pp. 145–151.

[20] R. Baly, H. Hajj, N. Habash, K. B. Shaban, and W. El-Hajj, "A sentiment treebank and morphologically enriched recursive deep models for effective sentiment analysis in arabic," *ACM Trans. Asian Low-Resource Lang. Inf. Process.*, vol. 16, no. 4, p. 23, 2017.

[21] M. Al-Smadi, O. Qawasmeh, M. Al-Ayyoub, Y. Jararweh, and B. Gupta, "Deep recurrent neural network vs. Support vector machine for aspect-based sentiment analysis of arabic hotels' reviews," *J. Comput. Sci.*, vol. 27, pp. 386–393, Jul. 2018.

[22] S. Boukil, M. Biniz, F. E. Adnani, L. Cherrat, and A. E. E. Moutaouakkil, "Arabic text classification using deep learning technics," *Int. J. Grid Distrib. Comput.*, vol. 11, no. 9, pp. 103–114, Sep. 2018.

[23] X. Chen, Y. Sun, B. Athiwaratkun, C. Cardie, and K. Weinberger, "Adversarial deep averaging networks for cross-lingual sentiment classification," *Trans. Assoc. Comput. Linguistics*, vol. 6, pp. 557–570, Dec. 2018.

[24] M. Alali, N. M. Sharef, M. A. A. Murad, H. Hamdan, and N. A. Husin, "Narrow convolutional neural network for arabic dialects polarity classification," *IEEE Access*, vol. 7, pp. 96272–96283, 2019.

[25] F. Altenberger and C. Lenz, "A non-technical survey on deep convolutional neural network architectures," 2018, *arXiv:1803.02129*. [Online]. Available: https://arxiv.org/abs/1803.02129

[26] A. Severyn and A. Moschitti, "Learning to rank short text pairs with convolutional deep neural networks," in *Proc. 38th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Aug. 2015, pp. 373–382.

[27] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep Learning*, vol. 1. Cambridge, MA, USA: MIT Press, 2016.

[28] J. Murphy, "An overview of convolutional neural network architectures for deep learning," Microway, Tech. Rep., 2016. [Online]. Available: https://www.microway.com/download/whitepaper/An_Overview_of_Convolutional_Neural_Network_Architectures_for_Deep_Learning_fall2016.pdf

[29] M. A. Nielsen, *Neural Networks and Deep Learning*, vol. 25. San Francisco, CA, USA: Determination Press USA, 2015.

[30] G. Tolias, R. Sicre, and H. Jégou, "Particular object retrieval with integral max-pooling of CNN activations," 2015, *arXiv:1511.05879*. [Online]. Available: https://arxiv.org/abs/1511.05879

[31] D. Whitley, "A genetic algorithm tutorial," *Statist. Comput.*, vol. 4, no. 2, pp. 65–85, Jun. 1994.

[32] M. Mitchell, "Genetic algorithms: An overview," *Complexity*, vol. 1, no. 1, pp. 31–39, Sep. 1995.
[33] K. Jebari and M. Madiafi, "Selection methods for genetic algorithms," *Int. J. Emerg. Sci.*, vol. 3, no. 4, pp. 333–344, Dec. 2013.
[34] Y. Yun and M. Gen, "Performance analysis of adaptive genetic algorithms with fuzzy logic and heuristics," *Fuzzy Optim. Decis. making*, vol. 2, no. 2, pp. 161–175, 2003.
[35] S. Haupt, *Practical Genetic Algorithms*. State College, PA, USA: Wiley, 2004, pp. 123–190.
[36] J. Carr, "An introduction to genetic algorithms," *Senior Project*, vol. 1, no. 40, p. 7, May 2014.
[37] A. Farghaly and K. Shaalan, "Arabic natural language processing: Challenges and solutions," *ACM Trans. Asian Lang. Inf. Process.*, vol. 8, no. 4, p. 14, Dec. 2009.
[38] J. J. Webster and C. Kit, "Tokenization as the initial phase in NLP," in *Proc. 14th Conf. Comput. Linguistics*, vol. 4, 1992, pp. 1106–1110.
[39] S. Qiu, M. Jiang, Z. Pei, and Y. Lu, "Text classification based on ReLU activation function of SAE algorithm," in *Proc. Int. Symp. Neural Netw.* Hokkaido, Japan: Springer, 2017, pp. 44–50.
[40] V. Daultani, Y. Ohno, and K. Ishizaka, "Sparse direct convolutional neural network," in *Advances in Neural Networks*. Cham, Switzerland: Springer, 2017, pp. 293–303.
[41] *Natural Language Toolkit*. Accessed: 2021. [Online]. Available: https://www.nltk.org/
[42] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1532–1543.

DEEM ALSALEH received the B.S. degree in computer science and the M.S. degree in software engineering from Prince Sultan University, Riyadh, Saudi Arabia. She is currently a Senior Software Engineer at Saudi Information Technology Company (SITE). She has published papers in international conferences and articles in scientific journals. Her research interests include machine learning, natural language processing, and software engineering.

SOUAD LARABI-MARIE-SAINTE received the Ph.D. degree in artificial intelligence from the Department of Computer Science, Toulouse1 University Capitole, France, in 2011, the Engineering degree in operation research from the University of Science and Technology Houari Boumediene, Algeria, the M.Sc. degree in mathematics, decision, and organization from Paris Dauphine University, France, and the M.Sc. degree in computing and applications from Sorbonne Paris1 University, France. She was an Associate Researcher with the Department of Computer Science, Toulouse1 University Capitole, and an Assistant Professor with the College of Computer and Information Sciences, King Saud University. She is currently an Associate Professor with the Department of Computer Science, the Associate Director of postgraduate programs with the College of Computer and Information Sciences, and the Vice-Chair of the ACM Professional Chapter, Prince Sultan University. She taught several courses at the graduate and postgraduate levels. She published several articles in ISI/Scopus indexed and attended various specialized international conferences. She is also an editorial board member and a reviewer of reputed journals and on the panel of TPC of international conferences. Her research interests include metaheuristics, artificial intelligence, healthcare, machine and deep learning, natural language processing, educational data mining, pattern recognition, and software engineering.