IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 4, 2020, accepted July 16, 2020, date of publication July 23, 2020, date of current version August 5, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3011508

Spark-Based Parallel Deep Neural Network Model for Classification of Large Scale RNAs into piRNAs and Non-piRNAs

SALMAN KHAN[1], MUKHTAJ KHAN[1], NADEEM IQBAL[1], (Senior Member, IEEE), MAOZHEN LI[2], AND DOST MUHAMMAD KHAN[1], (Member, IEEE)

[1]Department of Computer Science, Abdul Wali Khan University Mardan, Mardan 23200, Pakistan
[2]Department of Electronic and Computer Engineering, Brunel University London, London UB8 3PN, U.K.

Corresponding authors: Mukhtaj Khan (mukhtaj.khan@awkum.edu.pk) and Nadeem Iqbal (nikhan@awkum.edu.pk)

ABSTRACT With recent advancement in computational biology, high throughput next generation sequencing technology has become a de facto standard technology for genes expression studies including DNAs, RNAs and proteins. As a promising technology, it has significant impact on medical sciences and genomic research. However, it generates several millions of short DNA and RNA sequences with several petabytes size in single run. In addition, the raw sequencing datasets such as RNAs are increasing exponentially leading to a big data analytics issue in computational biology. Due to the explosive growth of RNA sequences, the timely classification of RNAs sequence into piRNAs and non-piRNAs have become a challenging issue for traditional technology and conventional machine learning algorithms. Parallel and distributed computing models along with deep neural network have become a major computing platform for big data analytics now required in the field of computational biology. This paper presents a computational model based on parallel deep neural network for timely classification of large number of RNAs sequence into piRNAs and non-piRNAs, taking advantages of parallel and distributed computing platform. The performance of the proposed model was extensively evaluated using two-fold performance metrics. In the first fold, the performance of the proposed model was assessed using accuracy-based metrics such as accuracy, specificity, sensitivity and Matthews's correlation coefficient. In the second fold, computational-based metrics such as computation times, speedup and scalability were observed. Moreover, initially the performance of the proposed model was assessed using real benchmark dataset and subsequently the performance was assessed using replicated benchmark dataset. The evaluation results in both cases showed that the proposed model improved computation speedup in order of magnitude in comparison with sequential approach without affected accuracy level.

INDEX TERMS Deep neural network, spark, big data, piRNA, classification algorithm, artificial intelligence.

I. INTRODUCTION

RNA is an important molecule in computational biology that stores genetic information embedded along a nucleic acid chain in the form of nucleotide bases series. The RNA molecules are grouped into coding RNA (cRNA) and non-coding RNA (ncRNA) molecules. The cRNA molecules are included mRNA (messenger RNA) which carries genetic information and actively involved in the process of translation of genes (DNA) into proteins. During the process of proteins synthesis, the mRNAs are relegated as a simple intermediate between genes and proteins. The ncRNA molecules have diverse grouped of RNA molecules including piRNAs (piwi interacting RNAs).

The piRNA molecule belongs to a larger class of small ncRNAs which is found in animal germline cells and human somatic cell. The piRNA molecules having a sequence length of 21 $\sim$ 35 nucleotides [1]–[4]. It is important molecule from the perspective of reproduction and development of germline cells and act as a guardian by protecting the germline cells from attack of transposable elements through transcriptional or post-transcriptional mechanisms [5] [6]. It has been

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

reported that the piRNA molecules play significant role in many genes functions such as translation of specific proteins, regulate gene expression, fight against viral infection, maintain genome integrity and transposon silencing [7]. Furthermore, the piRNA molecules move within genome and cause mutation, insertion and deletion that can lead to genome instability [8]. In addition, a number of studies have shown that the occurrence of piRNA is associated with multiple tumour types and a signature feature for the cancer cells development and progression. [9]–[11]. Hence, there is great interest in identification and classification of piRNA molecules for cancer cells diagnosis and therapy, drug development and genes stability.

With recent advancement in bioinformatics the raw sequencing datasets are increasing exponentially doubling in size every 18 months leading to a big data issue in computational biology [12]–[14]. In addition, a piRBase database has been constructed in 2014 which collected a comprehensive piRNA sequencing data. In 2014, the piRBase contained 77 million of piRNA sequences collected from 9 organisms. Recently, the number of unique piRNA sequences have reached to 173 million collected from 264 datasets from 21 species [15], [16]. This exponential growth in piRNA sequences lead to a huge amount of datasets which can be used in variety of applications including cancer diagnosis, drug discovery and precision medicine. However, the timely analysis and accurate classification of massive amounts of RNA sequences is not only becoming a challenging task for traditional machine learning algorithms but also enable difficulties in term of timely processing for a conventional technology.

In the literature a number of models have been developed for classification and prediction of piRNA sequences using machine learning (ML) algorithms, genetic algorithm and fisher discriminative method. For example, the work presented in [3], [17]–[19] proposed computational models using ML methods such as support vector machine (SVM) for identification of piRNA sequences. Zhang *et al.* [20] proposed a piRNA predictor using Fisher discriminant algorithm [21] with linear discriminant equation. Li *et al.* [22] build a classifier based on genetic algorithm for classification of RNA sequences into piRNA sequences and non-piRNA sequences.

The aforementioned models have produced promising results however; they have a number of limitations. Firstly, these models based on conventional ML methods with single stack processing layer which are unable to accurately classify piRNA sequences due to a high non-linearity in the sequences. Secondly, to extract optimal features for accurate classification, these models required significant amounts of human engineering and expertise. Thirdly, either the model based on non-deterministic algorithm (for example, genetic algorithm) which could not find an optimum solution or it based on linear equation (for example, Fisher algorithm) which does not support non-linearity in a dataset.

In order to mitigate the above limitations, an intelligent computational model such as DNN model can be utilized which apply multi-stack processing layers with weight optimization to implicitly extract optimum features from a given sequences using standard learning methods. Moreover, the DNN model effectively handles the non-linearity issue in the sequences by applying non-linear activation functions with multiple hidden layers [23], [24]. For this reason, the authors previously proposed a sequential 2L-piRNADNN [25] model based on multi-layer deep neural network that automatically extracts optimum features from a given feature vector using inherit features of DNN i.e. both the feature extraction and weight optimization are performed implicitly. The proposed 2L-piRNADNN model achieved a highest accuracy compared with other existing prediction models, however, it poses a huge computation cost and takes longer time during a training phase. The training time and computation cost of the model can be minimized through a parallel and distributed computing methodology.

A number of researchers have proposed parallel and distributed deep learning algorithms for various complex problems in order to reduce computation time. For instance, works presented in [26], [27] used distributed GPUs (graphic processing units) cluster to parallelize deep learning algorithms for speech recognition system. Similarly work presented in [28] proposed distributed DNN based on GPU cloud computing platform to address communication bottleneck issue arises during for data-parallel stochastic gradient descent (SGD). The mentioned models significantly reduced computation time; however, these models do not support a fault-tolerance feature, in case a GPU node failure occurred during the execution, the entire training process will be affected. In order to integrate the fault-tolerance feature, Sinthong *et al.* [29] parallelized DNN model using Hadoop MapReduce framework [30], [31] for video data analysis. Hadoop MapReduce is popular big data analytics framework offer built-in fault-tolerance feature, however, the Hadoop-based computational model performance is limited due a high I/O latency associated with the Hadoop framework [32]. Moreover, the high I/O latency limited the suitability of the Hadoop framework for iterative processing applications such machine learning algorithms.

In this paper, we proposed a scalable, fault-tolerant and parallel multi-layers DNN model for classification of massive amounts of RNAs sequence into piRNAs and non-piRNAs, taking advantages of distributed computing model. The proposed model is implemented using Spark framework [33] which has become a major computing platform for data intensive applications, making use of processing nodes of a cluster. Unlike the Hadoop, the Spark framework support in-memory computations where the data is stored and processed using shared memory during map phase and reduce phase computations. This feature minimizes I/O latency of the Spark framework and make it suitable computational framework for machine learning algorithms. Moreover, the proposed model implicitly applied data parallelization and code parallelization techniques that significantly reduced the DNN training time. Additionally, the proposed model applies dinucleotide

auto covariance method to formulate the RNA sequences into a feature vector of numeric values [34], [35].

The major contributions of the paper are as follow:

- Propose a scalable distributed deep generative model for classification of large-scale RNA sequences into piRNAs and non-piRNAs.
- The proposed model considered non-linearity in dataset using multi-stack processing layers and non-linear activation function.
- The proposed model implicitly distributes data and computations (i.e. model code) on number of processing nodes using spark framework to achieve massive parallelisms. Moreover, the fault-tolerance and scalability characteristics are integrated in the proposed model to make it a robust system.
- The performance of the proposed model is extensively evaluated using different performance measurement metrics such as accuracy, execution time, speedup and scalability.

The performance of the proposed model is evaluated in two stages. In the first stage, the real RNA sequences are used to assess the performance of the proposed model. In the second stage, the number of sequences is scale-up through replication process to generate a big data scenario. For the performance evaluation, we consider two sets of parameters such as (a) accuracy, specificity, sensitivity and Matthews's correlation coefficient and we called them accuracy-based metrics and (b) computation times, speedup and scalability and we called them computational-based metrics. The evaluation results showed that the proposed model has significantly reduced the training time and improved speedup more than 3x times using four processing nodes in comparison with sequential approach while maintained same level of accuracy as of the sequential approach.

The remainder of this paper is structured as follows. Section II provides background of deep neural network. Section III presents in details the architecture and implementation of the proposed model. Section IV presents experimental results and performance evaluation. Discussion is provided in Section V. Finally, Section VI provides concluding remarks and further research.

II. DEEP NEURAL NETWORK ARCHITECTURE

Neural Network is rapidly emerging as a powerful machine learning tool enabling high performance in accuracy and solved complex problems in bioinformatics. The main motivation behind the neural network was to develop a hierarchical network that enable to achieve intelligent behavior and perceived similar to the human neural system. Earlier, learning model such as perceptron [36], Adaline [37] are used linear model to train the input data. The linear model has various limitations such as the model was unable to solve the complex problem. Later, with the advancement in AI and bio-inspired models to solve non-linear problems and train deeper to leads the term deep neural network.

The deep learning algorithms allow computational models to have multi-stack processing layers through which a complex and non-linear functions can be easily learned. The deep learning methods have proved to be the most effective method in several fields, such as speech recognition [38], [39], image recognition [40]–[42], natural language processing [43] and bioinformatics [25], [44]–[46]. In addition, it has been stated in a number of publications that the deep learning models performed better than conventional machine learning algorithms for various complex learning issues [47]–[49].

Deep neural networks inspired from human brain activities and evolved from Neural Network having powerful learning ability to represent the big data in form of hierarchical representations. The learning ability of the DNN model is significantly improved in the recent years by considering different depths of the model i.e. multiple hidden layers with input and output layers linked through a free learning parameter such as weight. Furthermore, many researchers have investigated that the hierarchy structure of the DNN with increasing number of processing layers and data dimensionality leads to a computation complexity. Therefore, the DNN model using Spark computing platform is proposed in this paper in order to minimize computational complexity through a parallel processing.

In this paper, the DNN model is configured with 3-hidden layers with input layer and output layers as shown in Fig.1. Each layer has multiple neurons that process input features vector and produces output using Eq. (1). The weight matrix on every neuron is initialized using Xavier function [50] which has the ability to remain the variance same through each layer. Moreover, a backpropagation technique is applied to update the weight matrix in such a way that errors between the output class and target class are minimized. Nonlinear activation function i.e. sigmoid is applied at input layer and at hidden layers using Eq. (2). The activation function helps the model to learn non-linearity and complex patterns in

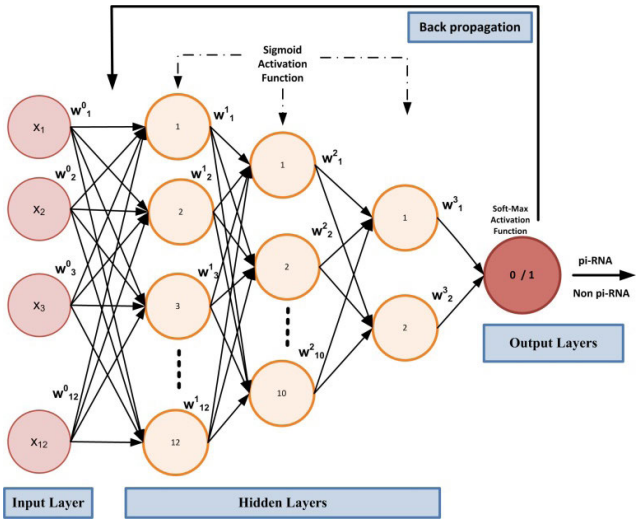

FIGURE 1. Architecture of deep neural network.

a dataset. Moreover, it determines either a neuron can be fired or ignored depending upon the output produced by that neuron [51]. Additionally, a softmax activation function is applied at output layer that generated a value in the range of [0,1] that represent the probability of data-point belong to a particular class.

$$y_i = g(b_i + \sum_{j=1}^{m} x_j w_j^i) \tag{1}$$

where y_i represent output at a layer i, b represent bias value, w_j^i represent weight used at a layer i by a neuron j, x_j^i represent input feature and g represent non-linear activation sigmoid function and it can be calculated using Eq. (2).

$$g(z) = \frac{1}{1 + e^{-z}} \tag{2}$$

As the DNN model computational cost and complexity increases with increasing the input size. Hence we analyzed the performance of the DNN model through calculating computational complexity of the algorithm. The computational complexity is calculated according to the size of input to the model i.e. n. The computational complexity reaches its least value when it is equal to $\Theta(1)$. As the number of input increases, ultimately, the computational complexity of the model is increased. The computational complexity of the proposed DNN can be divided into two main factors i.e. forward and back propagation [52]. In order to calculate the big O notation for the forward propagation we assume a fully connected DNN model having equal number of neurons and layers i.e. n having a time complexity is $\Theta\left(n^4\right)$ [52]. Furthermore for the back-propagation, the proposed network used gradient descent for n iterations, and that there are n layers each with n neurons, the total run-time of back-propagation is defined as $\Theta(n^5)$. Hence, the total computational complexity of the DNN algorithm is defined as

$$\Theta\left(n^5 + n^4\right) \cong (n^5) \tag{3}$$

Eq. (3) shows that the DNN algorithm retained the highest computational complexity in comparison with other learning algorithms. Hence the training time and computation cost of the DNN model can be minimized through a parallel and distributed computing methodology.

III. DESIGN OF PROPOSED MODEL

In this section we introduce design of the proposed model. The architecture of the proposed model is shown in Fig.3 which contains a number of components that are discussed in details as follow:

A. BENCHMARK DATASET

Benchmark dataset consists of piRNA and non-piRNA sequences. The benchmark dataset used in this paper is obtained from piRBase [16] which is a comprehensive database contained millions of piRNA sequences. We mathematically represent the benchmark dataset using Eq. (4).

$$D = D^+ \cup D^- \tag{4}$$

where D^+ represents the piRNA sequences and D^- represents the non-piRNA sequences. D represents union of both the piRNAs and the non-piRNAs sequence. We obtained 100 thousand of RNA sequences from the piRBase in which 50 thousand sequences were piRNA sequences (i.e. D^+) and the same numbers of sequences were non-piRNA sequences (i.e. D^-). Moreover, the benchmark dataset was divided into training dataset and testing dataset. The benchmark dataset i.e. 80% was used as the training dataset and remaining 20% of the benchmark dataset was used as the testing dataset.

B. FEATURE FORMULATION TECHNIQUE

In bioinformatics, biological sequences are originally represented in the form of FASTA format with various sequence length. However, the statistical machine learning methods are unable to process the biological sequences in the original form. These methods process the data represented in numeric values or discrete form [3], [25]. Therefore, it is required to formulate the biological sequences into a feature vector with numeric values before it given to a machine learning algorithm. However, pattern and order of the sequences information may be highly permutated during the process of formulation. A number of techniques have been introduced in the field of computational biology to formulate the RNA sequences with different sequence length into a feature vector without affecting pattern and order of the sequence information [53]. Additionally, a number of webservers have been established that formulate RNA sequences into a feature vector according to the user requirement [53]–[55].

In this paper, we have employed di-nucleotide auto covariance (DAC) method [34], [35] to transforms the RNA sequences of different lengths into a feature vector with uniform length. It measures correlation between two dinucleotide of same physiochemical property which is separated by a distance of *lag* along the sequence. Let suppose that a RNA sequence is represented by R with different nucleotides such as

$$R = R_1 R_2 R_3 \ldots\ldots\ldots R_L \tag{5}$$

where, $R_i \in \{A, C, G, U\}$ and $i(1, 2, 3, \ldots .L)$, R_1 represent a nucleotide at 1st position of the sequence, R_2 represent nucleotide at 2^{nd} position of the sequence and so forth. R_L represents a nucleotide at L position of the sequence. L represents the length of the RNA sequence. The alpha letter 'A', 'U', 'C' and 'G' represent Adenine, Uracil, Cytosine and Guanine respectively. Using the DAC method, the RNA sequence (i.e. represented in Eq. (5)) can be formulated into

a feature vector using Eq. (6).

$$\varphi(u, lag) = \frac{\sum_{i=1}^{L-lag-1} (P_u(R_iR_{i+1}) - \bar{P}_u)(P_u(R_{i+lag}R_{i+lag+1}) - \bar{P}_u)}{L - lag - 1} \tag{6}$$

where φ describes the correlations between two dinucleotide of the same physiochemical property, the indices of physicochemical properties ($u = 1,2,3,\ldots 6$) are denotes by u. ($P_u(R_iR_{i+1})$ ($P_u(R_{i+lag}R_{i+lag+1})$ represents the numerical value of index u for dinucleotide $R_iR_{i+1}(R_{i+lag}R_{i+lag+1})$ at location $i(k)$, $\overline{P_u}$ is the average value of physiochemical index u of entire sequence and can be calculated using Eq. (7).

$$\overline{P_u} = \sum_{k=1}^{L-1} \frac{P_u(R_kR_{k+1})}{L - 1} \tag{7}$$

We used Eq. (7) to extract the feature vector having length of LAG * N. Where LAG is the maximum of lag (i.e. LAG =2) and N represents the number of physicochemical properties. In this work we have considered six physicochemical properties. Further details of RNAs sequence formulation using DAC method is given in our previous publication [25].

It is to be noted that the sequence formulation module of the proposed model is sequential, and it may takes a high computation time in case it is applied on large number of sequences.

C. APACHE SPARK

Apache Spark is an open-source framework and distributed computing model known as an analytical engine for analyzing and processing large scale data using a cluster computing platform [56]. It is in-memory computation framework where the data is maintained and processed in shared physical memory. In case the memory is not enough and cannot store anymore data then the Spark spills out the data into secondary storage. This feature makes the Spark suitable computational framework for iterative algorithms such as machine learning algorithms. The Spark framework composed of SparkContext, driver program, cluster manager and worker node as shown in Fig.2. The SparkContext allows a user job to access cluster resources and allocates resources to a user job with the help of cluster manager. The driver program is a java process contained main method which generates SparkContext. The cluster manager is considered as an external module and responsible for resource management including resource allocation, scheduling and resources sharing across multiple jobs. The Spark framework can be deployed using different cluster mangers such as Standalone Cluster Manger, Hadoop Yarn and Mesos. The Spark framework is based on master/slave architecture. The master node is a central coordinator called Driver whereas the slave nodes are distributed

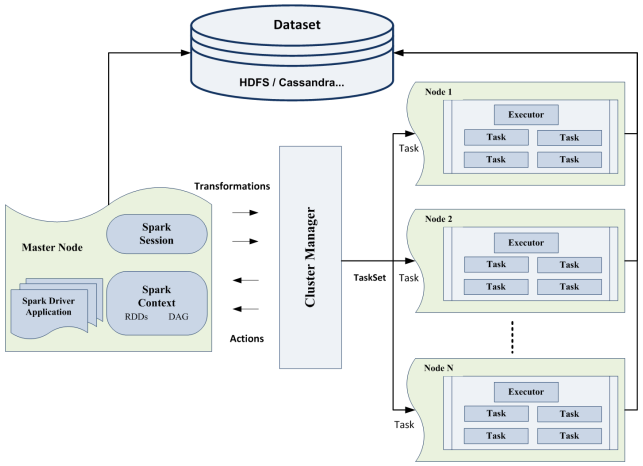

FIGURE 2. Execution mechanism of spark framework.

workers (executors). The Spark framework can access data from different sources such as HDFS, HBase, Cassandra and S3.

Additionally, a number of built-in libraries such as Spark SQL, MLib, GraphX, and Spark Streaming are included in the Spark framework to facilitate application developers in different domains. The Spark SQL allows application developers to query structure data using Spark system. The MLib is a scalable library for machine learning algorithms such as SVM, K-nearest neighbor, random forest, etc. GraphX is built-in library designed for parallel graphic iterative computations. The Spark Streaming is language-integrated API that facilitates developers to quickly create scalable, fault-tolerant applications for streaming and real-time data processing.

In addition to the built-in libraries, a significant memory abstraction introduced in the Spark framework is the in-memory resilient distributed dataset (RDD), which provides high scalability and fault tolerance capabilities [57]. The RDD is a set of read-only objects that partitioned across several cluster processing nodes to allow parallel processing. Moreover, the RDD achieve a fault-tolerance capability through implementation of RDD Lineage [58] service. In case a node failure is occurred during execution, the RDD lineage automatically recalculate the lost RDD partition from its parent RDD.

D. PARALLEL DEEP NEURAL NETWORK

In this section, we introduce parallel deep neural network using Spark framework. The parallel module of the proposed model is shown in Fig.3 where the Spark framework divide large training data into samples of RDDs, denoted as D_1, D_2, D_3,...,D_n and distributed across a cluster of nodes (i.e. using data parallelization). In addition, the Spark framework distributes a copy of the DNN model across multiple workers (i.e. using model parallelization). The training process is started by executing the DNN model across the worker nodes simultaneously and multiple models are

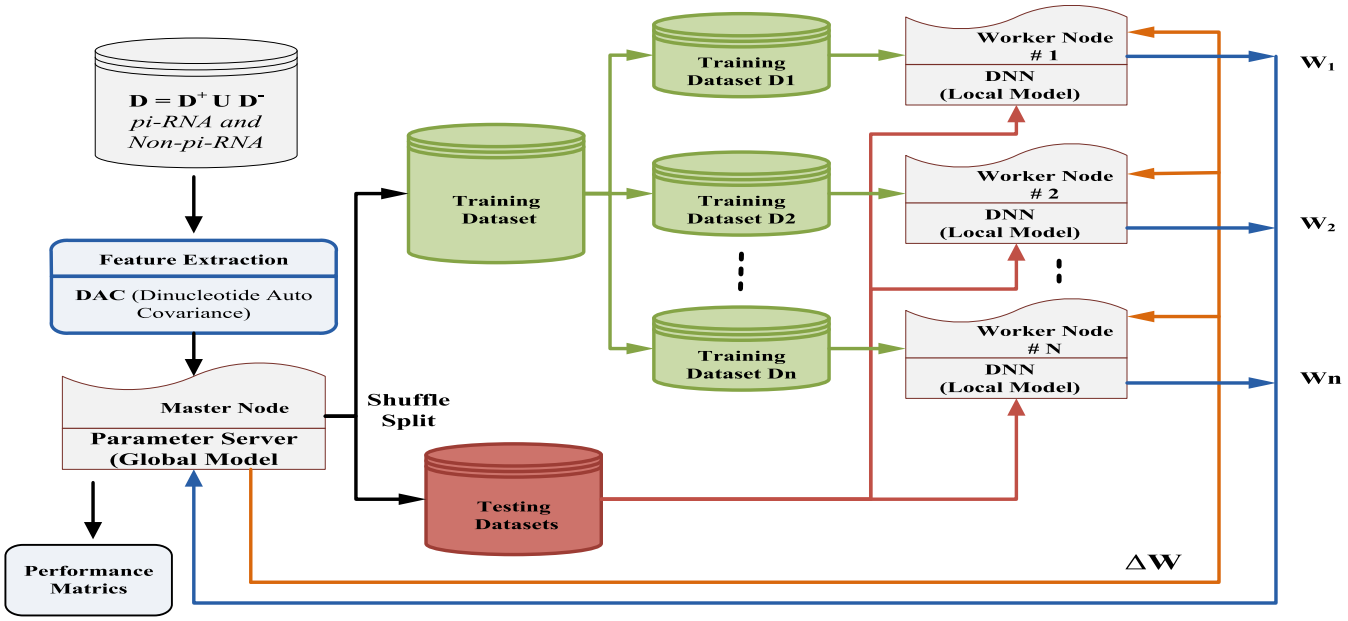

FIGURE 3. Framework of proposed model. The blue color lines show that each worker node updates the parameter server with their local model (i.e. W_i). The orange lines show that the parameters server share the new weight (i.e. ΔW) with the worker nodes.

trained in parallel with given hyper-parameters as listed in Table 2. At the end of the training process, each worker node updates the master node with the trained models (represented with blue line in Fig.3). After a global trained model is achieved, each worker is given a separated test dataset as samples of RDD. The worker nodes apply the global trained model on the given test dataset and generate a local result (classification metrics). Finally, the n local results are combined through a master node (parameter server) to produce a final result by applying average parameter function.

In order to optimize the proposed model, we apply back-propagation technique in each model deployed on the worker nodes. In every iteration, the DNN model goes back and forth to minimize the loss function. Each worker node computes SGD on a given subset to minimize prediction error. Each worker node reports the computed SGD to a centralized parameter server. Synchronous communication mechanism is applied for communication between the worker nodes and the parameter server. The parameter server collects partial gradients from all the worker nodes and computes a set of new weights. Then the parameter server shared the new weights with the worker nodes to perform gradient computation again (represented in orange line in Fig.3). In the proposed model, the server parameter manages the distributed scaling model and serves as a coordination agent between the main server and the worker nodes. [59]. The synchronous SGD achieve a better efficiency and scalability compared to asynchronous SGD optimization [60], [61], however, the synchronous SGD operation may degrade the performance of the model due to occasionally slowdowns a worker node. In the proposed model, we assume that all the worker nodes are homogenous and there is no slowdown worker node. At high level, the proposed model performs training and testing as:

1. Spark framework divides large number of sequences into small partitions and distribute across all worker nodes.
2. A copy of DNN model is replicated on all the worker nodes.
3. Each worker node executes the model on given partition to train a local model.
4. The worker node reports parameters of the local model to a parameter server. The parameter server aggregates the parameters to obtain a global model by averaging of all the local models parameters.
5. The parameter server updates the weights and shares new weight with the worker nodes.
6. The proposed model repeats steps 3, 4 and 5 several times to obtain a global trained model with optimized parameters.
7. The global trained model is then applied on testing dataset to generate final average classification metrics.

The proposed approach significantly reduces computation time with a high scalability using parallel approach. The proposed model can be scaled up by adding more nodes that further improves the performance of the model in term of computation times and speedup (discussed in detail in section IV). Additionally, the proposed model achieved fault-tolerance feature by sharing multiple copies of data samples in the form of RDDs across several worker nodes which overcome the issue of a node failure situation. In case a node failure is occurred during a job execution, the spark framework automatically detects the node failure and assigns

workload of the failed node to another available node without affects the job completion.

E. PERFORMANCE EVALUATION METRICS

The performance of the proposed model was assessed using accuracy-based metrics and computational-based metrics. For the accuracy-based metrics, we considered the widely used metrics such as: (i) ACC, reflect the overall accuracy of a model (ii) SP, represent a model precision, (iii) SN, represent a model's sensitivity and (iv), MCC, represent a model Mathew's correlation coefficient [62]. These metrics can be calculated using Eq. (8) – (11).

$$ACC = 1 - \frac{N_{-}^{+} + N_{+}^{-}}{N^{+} + N^{-}}, 0 \leq Acc \leq 1 \tag{8}$$

$$SP = 1 - \frac{N_{+}^{-}}{N^{-}}, 0 \leq Sp \leq 1 \tag{9}$$

$$SN = 1 - \frac{N_{-}^{+}}{N^{+}}, 0 \leq Sn \leq 1 \tag{10}$$

$$MCC = \frac{1 - \left(\frac{N_{-}^{+} + N_{+}^{-}}{N^{+} + N^{-}}\right)}{\sqrt{\left(1 + \frac{N_{+}^{-} - N_{-}^{+}}{N^{+}}\right)\left(1 + \frac{N_{-}^{+} - N_{+}^{-}}{N^{-}}\right)}}, \quad -1 \leq Mcc \leq 1 \tag{11}$$

where

- N^{+} represents the total number of piRNA sequences.
- N^{-} represents the total number of non-piRNA sequences.
- N_{-}^{+} represents the total number of piRNA sequences wrongly predicted by the proposed model as non-piRNA sequences
- N_{+}^{-} represents the total non-piRNA sequences incorrectly predicted by the proposed model as piRNA sequences.

The computational-based metrics such as computation time, speedup and scalability are analytical metrics and can be achieved through simulation results. However, the speedup of parallel processing can be calculated using Eq. (12).

IV. RESULTS AND DISCUSSION

In this section we discuss and evaluate the performance and efficiency of the proposed model using both the accuracy-based metrics and the computational-based metrics as mentioned in section III (E). The performance of the proposed model was initially assessed using the benchmark dataset and then subsequently assessed using replicated dataset having large number of sequences.

A. EXPERIMENTAL SETUP

We have setup a Spark cluster with default configurations using four physical processing nodes. Table 1 describes the basic hardware and software specifications used in the experiments. Ubuntu 12.04 LTS operating system along with Spark 2.0 and Hadoop 2.7.3 were configured on all processing nodes. One processing node was configured as master node and the remaining three nodes were configured as worker nodes. Moreover, the master node was also used as a worker node.

TABLE 1. Configuration detail of apache spark cluster.

Single Node Specification	CPU	Intel Core Tm
	Processor	3.20 GHz x 4
	Hard disk	480.4 GB
	Connectivity	100Mbps Ethernet LAN
	Memory	16 GB
Software	Operating System	Ubuntu 12.04 LTS
	JDK	1.8
	Hadoop	2.7.3
	Spark	2.0
	OS Type	64-bit

TABLE 2. List of Hyper-parameters of DNN model with optimum values.

Parameters	Optimized Values
Iteration	400
Learning	0.1
Activation Function	Sigmoid
Seed	1234L
Number of hidden layers	3
Number of Neurons	12-10-2-1
Weight initialization	XAVIER function
Regularization.l2	0.001
Dropout	0.25
Optimizer	SGD Method
Updater	ADAGRAD function

B. DNN PARAMETERS OPTIMIZATION

Deep learning network topology are usually involved a number of parameters that greatly impact on the performance of a model. These parameters are listed in Table 2 and referred to as hyper-parameters. Grid search technique was applied to find a set of optimum hyper-parameter by evaluating the outcome of the model on different set of the hyper-parameter values using benchmark dataset. A number of experiments were performed to find the optimum values for activation function, learning rate, number of hidden layers and number of neurons in each hidden layer. However, we present the impact of only two highly influential hyper-parameters such as learning rate and activation function in Table 3. The table shows that how the learning rate and the activation functions are significantly impacted on the outcome of the model.

It can be observed from the table that the model accuracy is slightly increased with decreasing the learning rates.

TABLE 3. Impact of different learning rate and activation function on accuracy of proposed model.

Learning Rate	Sigmoid Accuracy (%)	Tanh Accuracy (%)
0.08	81.74	74.13
0.09	81.74	74.13
0.1	81.74	74.13
0.2	81.71	74.12
0.3	81.70	74.10
0.4	81.68	74.09
0.5	81.67	74.07
0.6	81.65	74.06
0.7	81.63	74.04
0.8	81.61	74.03
0.9	81.60	74.01

For examples, at learning rate 0.9, the DNN model achieved a maximum accuracy of 81.60functions respectively. On the other side, at learning rate 0.1, the model achieved a highest accuracy 81.74% and 74.13% using sigmoid and Tanh activation functions respectively. Moreover, the learning rate was further decreased to 0.08; however, the overall accuracy of the proposed model was not significantly changed as shown in Table. As we can see from Table 3 that the proposed model achieved the highest accuracy using sigmoid activation function with learning rate 0.1. Hence, the optimum configuration values for both the learning rate and activation function are 0.1 and sigmoid respectively. The hyper-parameters along with optimum values found through grid search are presented in Table 2.

C. PERFORMANCE COMPARISON OF LEARNING ALGORITHMS

In this section we compare performance of proposed DNN with other widely used machine learning algorithms such as support vector machine (SVM) [63], K-nearest neighbor (KNN) [64] and random forest (RF) [65]. In order to ensure a fair comparison, we used same benchmark dataset, same experimental setup and same accuracy-based measurement metrics. Additionally we used optimized parameters for all the learning algorithms during the performance comparison. The parameters along with optimized values are presented in Table 2 and Table 4. Fig.4 shows performance comparison of the different learning algorithms. The figure illustrated that the DNN model performed better than other the machine learning algorithms. For example, the DNN model generated a highest accuracy i.e. 81.74% whereas the RF produced a second highest accuracy i.e. 68.98. The lowest accuracy generated by the KNN model which is 61.07.

The main reason of outstanding performance of the DNN model due to the multi-stack processing layers (i.e. hidden layers) with implicit weight optimization (i.e. backpropagation) which effectively handled a complex nature dataset having a high non-linearity whereas, the other learning models based on single-stack processing layer which is not sufficient to handle effectively a dataset with a high non-linearity. As the DNN model generated a highest accuracy and performance, therefore, next we have only considered the DNN model for further evaluation and parallelization.

TABLE 4. List of parameters of traditional machine learning algorithms with optimized values.

Classifier	Parameters	Optimized Values
SVM	Kernal Type	RBF
	Kernal Degree	2
	Cost	0.1
	Gamma	0.001
	Coef0	9
	Shrinkage	TRUE
RF	No. of trees	350
	Mtry	150
KNN	k-neighbors	500
	Weighting	Similarity

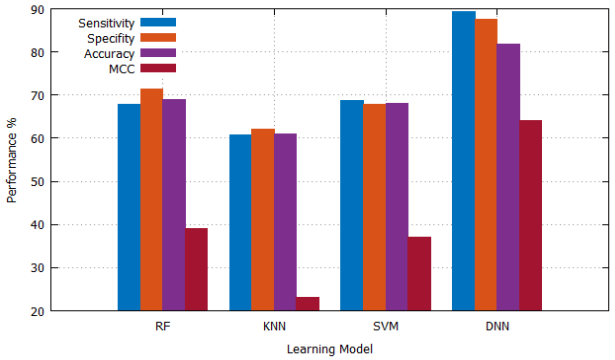

FIGURE 4. Performance comparison of machine learning algorithms.

D. PERFORMANCE EVALUATION OF DNN MODEL USING BENCHMARK DATA SET

We further extensively evaluate the performance of the DNN model using different performance measurement metrics. For this evaluation, the benchmark dataset was split into subsets with different number of sequences in thousands (i.e. 20,40,60,80,100). Each subset was contained equal number of piRNA and non-piRNA sequences.

Firstly, the performance of the proposed model was analyzed using accuracy-based metrics using different number of sequences. The result of this evaluation is reported in Table 5. Additionally, the accuracy of the proposed approach in comparison with sequential approach is illustrated in Fig.5. We can observe from Table 5 that both the approaches (i.e. parallel and sequential) produced all most similar values for

TABLE 5. Performance of proposed model in comparison with sequential model using varied number of sequences.

Classification Algorithms	SN (%)	SP (%)	ACC (%)	MCC
Sequential				
20 K	89.2485	87.5304	81.7366	0.6426
40 K	89.2167	87.5756	81.7367	0.6427
60 K	89.2582	87.5907	81.7369	0.6429
80 K	89.2590	87.5919	81.7372	0.6429
100 K	89.2623	87.5932	81.7374	0.6430
Parallel				
20 K	89.2485	87.5233	81.7365	0.6426
40 K	89.2167	87.5610	81.7366	0.6426
60 K	89.2582	87.5842	81.7368	0.6427
80 K	89.2590	87.5917	81.7371	0.6430
100 K	89.2596	87.5931	81.7374	0.6430

the accuracy-based metrics. However, it can be observed that in both cases the performance of the models is slightly improved with increased number of sequences. Moreover, Fig. 5, shows comparison of relative accuracies achieved both the approaches with varied number of sequences. When we compare the relative accuracy of the proposed model with sequential approach, it is important to note that there is a possibility of difference in the accuracy caused by the sequences partition due to the way in which the sequences are divided up for parallelization purpose. As we can see in Fig.5 that the accuracy difference between the sequential approach and the parallel approach is very small and can be neglected. For example, the sequential approach generated the accuracy of 81.7366% whereas the proposed model is generated the accuracy of 81.7365% using 20M sequences. The accuracy difference between the two approaches is 0.0001% which can be ignored during the classification process. Additionally, we can see that the accuracy difference is converging to zero with increased number of sequences.

Secondly, the performance of the parallel proposed model was evaluated using computational-based metrics such as computation time, scalability and speedup. First we compare the execution times of the proposed model with the sequential model using different number of sequences. The experimental results of this comparison are presented in Fig. 6. It can be clearly observed from the figure that the execution times of the proposed model are significantly reduced using four physical processing nodes compared with sequential approach. The execution time of the sequential approach is clearly increased with increasing number of sequences, whereas, the execution time of the proposed model is slightly increased with increasing number of sequences.

Second, we evaluate scalability of the proposed model using both a varied number of sequences and different

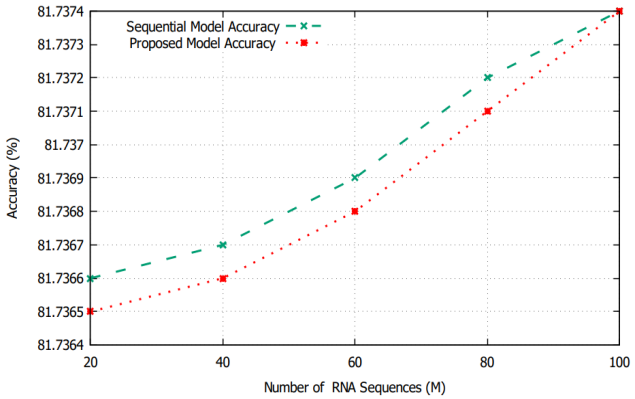

FIGURE 5. Accuracy comparison of proposed model with sequential approach using different number of sequences of real benchmark dataset.

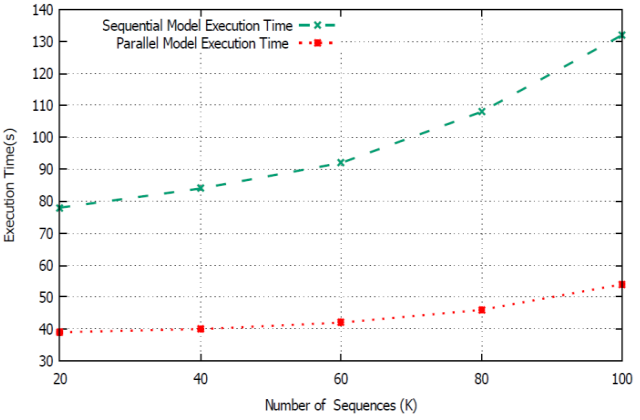

FIGURE 6. Efficiency analysis of the proposed model in term of execution times using real benchmark dataset.

number of processing nodes. The results of this evaluation are shown in Fig.7. The figure shows the execution times of the proposed model when it processed a varied number of sequences using processing nodes from 1 to 4. We can see from Fig.7 that execution time of the proposed model significantly decreased with increasing number of processing nodes employed. For example, the proposed model took 132 seconds when it classified 100 thousand RNA sequences into piRNAs and non-piRNAs using single processing node whereas the execution time of the proposed model dropped to 54 seconds when it classified the same number of sequences using four processing nodes. These results imply that the proposed model achieved 41% reduction in execution time when it processed a large number of sequences using four processing nodes.

Next we analyze the speedup of the proposed model using a varied number of sequences and processing nodes. The speedup of the proposed model was calculated using Eq. (12).

$$S = \frac{T_1}{T_n} \tag{12}$$

where S represent speedup of the proposed model. T_1 represent execution time of the proposed model on single

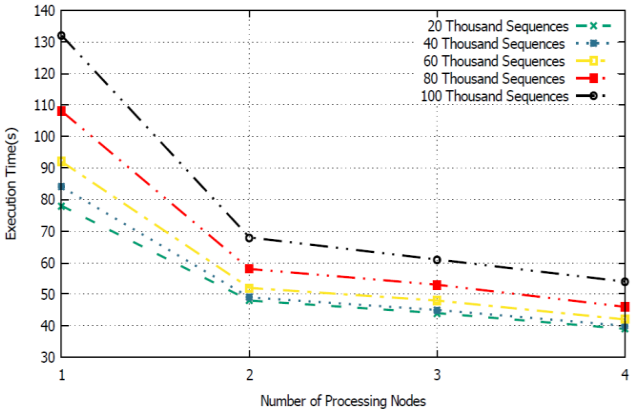

FIGURE 7. Scalability analysis of the proposed model using different number of processing nodes and varied number of sequences of real benchmark dataset.

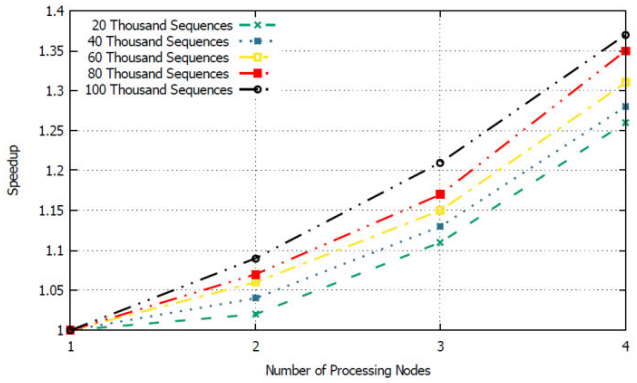

FIGURE 8. Speedup analysis of the proposed model on different processing nodes and varied number of sequences of real benchmark dataset.

processing node. T_n represent the execution time of the proposed model using n number of processing nodes (in our case $n = 4$). The result of Eq. (12) is demonstrated in Fig.8 which shows the speedup of the proposed model. In Fig.8, when the proposed model processed 80 thousand of sequences using 2 processing nodes, it generated 1.07 times speedup, whereas, using 4 processing nodes it generated 1.35 times speedup on the same number of sequences. However, when the number of sequences was increased up to 100 thousand, 2 processing nodes generated 1.09 times speedup, whereas, 4 processing nodes generated 1.37 times speedup. These results imply that in both cases i.e. increasing number of sequences and increasing the number of processing nodes, the speedup of the proposed model improved.

E. PERFORMANCE EVALUATION USING REPLICATED DATASET

In this section we assess the performance of the proposed model using replicated sequences. It is to be noted that the feature formulation module (i.e. section III (B)) of the proposed model applies sequential approach to formulate RNA sequences into a feature vector. It takes longer times

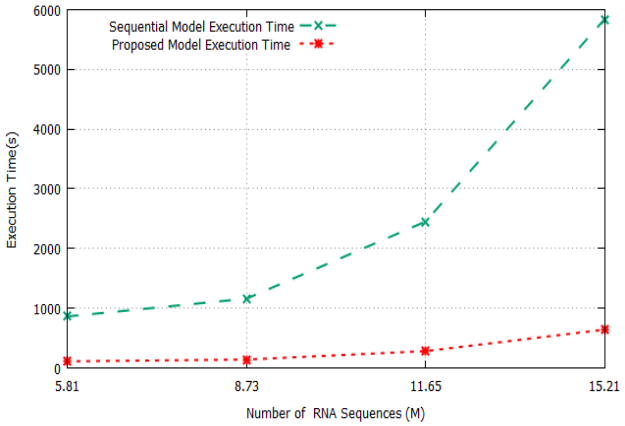

FIGURE 9. Efficiency analysis of proposed model in term of execution times using replicate benchmark dataset.

to formulate a large number of sequences into a feature vector. In order to evaluate the performance of the proposed model in big data scenario, we replicated the feature vector (formulated sequence) multiple times to generate a feature vector with high dimensionality resultant a big data scenario. As the dataset contained redundant and duplicate sequences, therefore, evaluating the performance of the proposed model using the accuracy-based metrics is meaningless. Therefore, in this evaluation we consider only the computational-based metrics to assess the performance of the proposed model.

Efficiency of the proposed model in comparison with sequential approach is illustrated in Fig.9. We can see from Fig.9 that the execution time of sequential approach is exponentially increased with increasing number of sequences whereas, the execution time of the proposed model is slightly increased with increased number of sequences. Moreover, the time difference between the sequential approach and the parallel approach is increased with increasing the number of sequences. This is because the sequential approach is unable to properly handle large number of sequences within reasonable time. On the other side, the proposed model can process large number of sequences within reasonable time.

The scalability analysis of the proposed model in term of both with a varied number of sequences and different number of processing nodes are shown in Fig.10. It is clearly shown in the figures that the proposed model execution time is greatly reduced with increasing number of processing nodes. For example, the execution time of the proposed model on single machine is 2127 seconds when it classified 15.21 million sequences whereas the execution time is dropped to 641 seconds using four processing nodes when it classified the same number of sequences. These results imply that the proposed model achieved a 30% reduction in execution time on a large number of sequences compared with single machine execution time.

The speedup of the proposed model was calculated using Eq. (12) when it processed four different sizes of sequences.

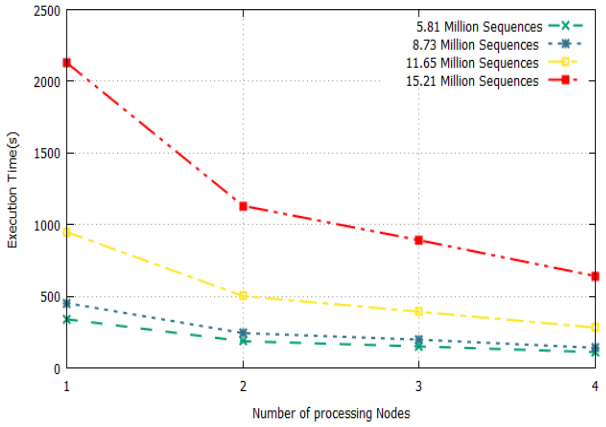

FIGURE 10. Scalability analysis of proposed model using different number of processing nodes and varied number of sequences of replicated benchmark dataset.

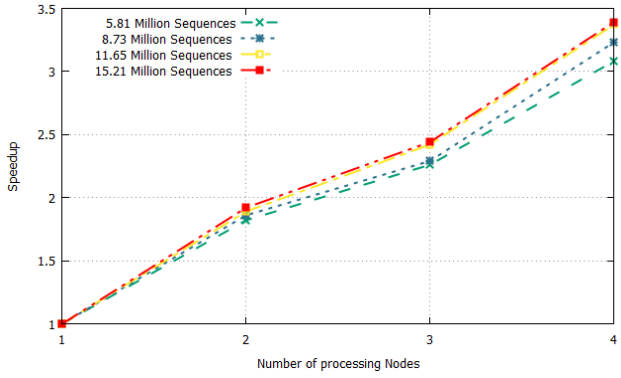

FIGURE 11. Speedup analysis of the proposed model on different number processing nodes and varied number sequences of replicated benchmark dataset.

The result of Eq. (12) is demonstrated in Fig.11. Consider Fig. 11, the proposed model achieved 1.82 times speedup on 2 processing nodes and 3.08 times speedup on 4 processing nodes when it processed 5.81 (M) sequences. Whereas, the proposed model achieved 1.92 times speedup on 2 processing nodes and 3.39 times speedup on 4 processing nodes when it processed 15.21 (M) sequences. These results confirm that the proposed model is highly scalable in term of both dataset size and number of processing nodes.

However, the proposed model never achieved the speedup up to the number of processing nodes which are to be expected from Amdahl's law [66]. This is due to a number of factors includingcluster communication overhead, task initialization overhead and network bandwidth. This phenomena is discussed in detail in [67]

Finally, we analyze speedup of the proposed model in both cases i.e. small number of sequences (i.e. real benchmark dataset) and large number of sequences (i.e. replicated benchmark dataset). In the case of real benchmark dataset, the proposed model gained a maximum speedup 1.37 times on 4 processing nodes when it processed 100 (K) sequences. Whereas, in the case of replicated benchmark dataset, the proposed model gained a maximum speedup 3.39 times on 4 processing nodes when it processed 15.21 (M) sequences. These result shows that the proposed model achieved a maximum performance (i.e. speedup) while processed large number of sequences. This analysis also confirms that the proposed model support processing of large number of sequences with high scalability.

V. DISCUSSION

piRNAs are small sequence of non-coding RNAs that provides several functions related to the protection and regulation of genes and genome. However, the complex structure of piRNAs sequence due to a high non-linearity exist, their accurate identification for traditional learning algorithms has become a challenging task. A DNN model which provides reasonable solutions for a complex problem has been employed in this research for classification of piRNAs and non-piRNAs sequence. As piRNAs sequence increases in size and DNN in complexity, computational intensity and memory demand increases proportionally. A high performance computing cluster is essentially required for training a DNN model to an adequate level of accuracy within reasonable time. In this paper we proposed a robust parallel and distributed DNN model using Spark paradigm for classification of massive RNAs sequence into piRNA and non-piRNA.

The DNN model is configured with several hyper-parameters which significantly effect on the performance of the model. A grid search technique was employed to search optimum hyper-parameters such as activation function, learning rate, number of hidden layers and neurons. Additionally, most of parallel and distributed algorithms based on GPUs or within box are facing different issues such as fault-tolerance, data availability and scalability. The proposed model achieved the fault-tolerance through heartbeat message whereas the data availability gained through replication of several copies of dataset on multiple machines. Furthermore, the proposed model is highly scalable and can be scaled better for larger problems / models. It support scale-out, means that the proposed model can be parallelized on any number of machines depends on size of the problem and computation complexity.

In order to mature the experimental results, the performance of the proposed model was rigorously evaluated using accuracy-based metrics and computational-based metrics. For example, Fig. 4 showed that DNN model outperformed all other machine learning algorithms using various performance measurement metrics. The main reason of better performance of the proposed DNN due to multi-stack processing layers with standard learning methods and weight optimization. The computation times of the proposed model was significantly reduced as increased number of processing nodes, shown in Fig.6 and Fig.9. We have noticed that the parallelization approach slightly affected the model accuracy using small number of sequences, however, the accuracy of the model was improved and difference between sequential and parallel models were converge to zero when increased

the number of sequences as shown in Fig. 5. Additionally we evaluated the speedup of the proposed model on number of processing machines as reported in Fig. 8 and Fig. 11. From these figure we can see that the proposed model achieved different speedup on different number of sequences using 4 processing nodes. For example, in case of small number of sequences (i.e. 1 M), the proposed model achieved a maximum speed up i.e. 1.37 which is far below from the maximum speed up i.e. 3.37 achieved in the case of processing large number of sequences (i.e. 15.21 M). These finding shows that the proposed model performed better on large number of sequences compared with small number of sequences. Furthermore, these figures show that the proposed model could not achieved a maximum speed up (i.e. equal to the number of processing nodes) in any case, this is due to tasks initialization overhead and network cognition occurred during a data shuffle phase.

As the DNN model exhibited promising results however a number of limitations are associated with the model: a) the model is sensitive to the number of neurons in the hidden layers i.e. a few neurons may cause under-fitting problems whereas too many neurons may contribute to an over-fitting problem. b) In general, a network with more hidden layers in a training or testing process can lead to a better accuracy, however, it arise major issues such as computation cost, complexity and vanishing gradient. c) Due a high computational complexity, the model required high performance computing to generate reasonable results within time. d) The model required a considerable amounts of training data to acquire effective results (ref. Fig. 5) [68]. To address these issues we applied a number of techniques. For example, a dropout technique was employed at each hidden layer to overcome the issue of over-fitting [38]. Similarly, a balanced number of hidden layers (i.e. 3 hidden layers) was configure in the network to overcome the computational complexity and vanishing gradient issue. Additionally, a distributed computing methodology was employed to overwhelmed computational complexity issue.

VI. CONCLUSION AND FUTURE WORK

This paper presented a parallel deep neural network model for classification of massive RNA sequences into piRNA sequences and non-piRNA sequences. The proposed model was built using Spark programming model to achieve a parallel computation by partitioning and distributing of sequences amongst a cluster of computer nodes. In addition, the proposed model applied DAC method for the sequence formulation. The performance of the proposed model was extensively assessed and experimental results showed that the proposed model achieved a high computation speedup in order of magnitude due to parallelization of both data and model in comparison with the sequential approach without affected the model accuracy. Additionally, the experimental results showed that the proposed model is highly scalable in term of both dataset size and number of processing nodes.

The proposed model was implemented with default parameters settings. The Spark framework have several configures parameters that significantly effects the performance of the framework. In future, we have planned to propose a methodology that automatically optimizes the configuration parameter of the framework using gene expression programming [69] and particle swarm optimization technique [70]. This will further improve the performance of the proposed model in term of computation speedup.

REFERENCES

[1] J. Cheng, H. Deng, B. Xiao, H. Zhou, F. Zhou, Z. Shen, and J. Guo, "PiR-823, a novel non-coding small RNA, demonstrates *in vitro* and *in vivo* tumor suppressive activity in human gastric cancer cells," *Cancer Lett.*, vol. 315, no. 1, pp. 12–17, Feb. 2012.

[2] A. Aravin, D. Gaidatzis, S. Pfeffer, M. Lagos-Quintana, P. Landgraf, N. Iovino, P. Morris, M. J. Brownstein, S. Kuramochi-Miyagawa, T. Nakano, M. Chien, J. J. Russo, J. Ju, R. Sheridan, C. Sander, M. Zavolan, and T. Tuschl, "A novel class of small RNAs bind to MILI protein in mouse testes," *Nature*, vol. 442, no. 7099, pp. 203–207, Jul. 2006.

[3] B. Liu, F. Yang, and K.-C. Chou, "2L-piRNA: A two-layer ensemble classifier for identifying PIWI-interacting RNAs and their function," *Mol. Therapy-Nucleic Acids*, vol. 7, pp. 267–277, Jun. 2017.

[4] D. M. Ozata, I. Gainetdinov, A. Zoch, D. O'Carroll, and P. D. Zamore, "PIWI-interacting RNAs: Small RNAs with big functions," *Nature Rev. Genet.*, vol. 20, no. 2, pp. 89–108, Feb. 2019.

[5] S. Choudhuri, "Epigenetic regulation of gene and genome expression," in *Reproductive and Developmental Toxicology*, R. C. Gupta, Ed. San Diego, CA, USA: Academic, 2011, ch. 60, pp. 801–813.

[6] Y. W. Iwasaki, M. C. Siomi, and H. Siomi, "PIWI-interacting RNA: Its biogenesis and functions," *Annu. Rev. Biochem.*, vol. 84, no. 1, pp. 405–433, Jun. 2015.

[7] C. Klattenhoff and W. Theurkauf, "Biogenesis and germline functions of piRNAs," *Development*, vol. 135, no. 1, pp. 3–9, Nov. 2007.

[8] S. Houwing, L. M. Kamminga, E. Berezikov, D. Cronembold, A. Girard, H. van den Elst, D. V. Filippov, H. Blaser, E. Raz, C. B. Moens, R. H. A. Plasterk, G. J. Hannon, B. W. Draper, and R. F. Ketting, "A role for piwi and piRNAs in germ cell maintenance and transposon silencing in zebrafish," *Cell*, vol. 129, no. 1, pp. 69–82, Apr. 2007.

[9] K. W. Ng, C. Anderson, E. A. Marshall, B. C. Minatel, K. S. S. Enfield, H. L. Saprunoff, W. L. Lam, and V. D. Martinez, "PIWI-interacting RNAs in cancer: Emerging functions and clinical utility," *Mol. Cancer*, vol. 15, no. 1, Dec. 2016, Art. no. 5.

[10] M. Moyano and G. Stefani, "PiRNA involvement in genome stability and human cancer," *J. Hematol. Oncol.*, vol. 8, no. 1, p. 38, Dec. 2015.

[11] J. Cheng, J.-M. Guo, B.-X. Xiao, Y. Miao, Z. Jiang, H. Zhou, and Q.-N. Li, "PiRNA, the new non-coding RNA, is aberrantly expressed in human cancer cells," *Clin. Chim. Acta*, vol. 412, nos. 17–18, pp. 1621–1625, Aug. 2011.

[12] B. Schmidt and A. Hildebrandt, "Next-generation sequencing: Big data meets high performance computing," *Drug Discovery Today*, vol. 22, no. 4, pp. 712–717, Apr. 2017.

[13] B. Langmead and A. Nellore, "Cloud computing for genomic data analysis and collaboration," *Nature Rev. Genet.*, vol. 19, no. 4, pp. 208–219, Apr. 2018.

[14] G. Zararsiz, D. Goksuluk, S. Korkmaz, V. Eldem, I. P. Duru, T. Unver, and A. Ozturk, "Classification of RNA-seq data via bagging support vector machines," *bioRxiv*, 2014.

[15] P. Zhang, X. Si, G. Skogerbø, J. Wang, D. Cui, Y. Li, X. Sun, L. Liu, B. Sun, R. Chen, S. He, and D.-W. Huang, "PiRBase: A Web resource assisting piRNA functional study," *Database*, vol. 2014, pp. 1–7, Jan. 2014.

[16] J. Wang, P. Zhang, Y. Lu, Y. Li, Y. Zheng, Y. Kan, R. Chen, and S. He, "PiRBase: A comprehensive database of piRNA sequences," *Nucleic Acids Res.*, vol. 47, no. D1, pp. D175–D180, Jan. 2019.

[17] K. Wang, C. Liang, J. Liu, H. Xiao, S. Huang, J. Xu, and F. Li, "Prediction of piRNAs using transposon interaction and a support vector machine," *BMC Bioinf.*, vol. 15, no. 1, p. 419, Dec. 2014.

[18] L. Luo, D. Li, W. Zhang, S. Tu, X. Zhu, and G. Tian, "Accurate prediction of transposon-derived piRNAs by integrating various sequential and physicochemical features," *PLoS ONE*, vol. 11, no. 4, Apr. 2016, Art. no. e0153268.

[19] T. Li, M. Gao, R. Song, Q. Yin, and Y. Chen, "Support vector machine classifier for accurate identification of piRNA," *Appl. Sci.*, vol. 8, no. 11, p. 2204, Nov. 2018.

[20] Y. Zhang, X. Wang, and L. Kang, "A k-mer scheme to predict piRNAs and characterize locust piRNAs," *Bioinformatics*, vol. 27, no. 6, pp. 771–776, Mar. 2011.

[21] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Ann. Eugenics*, vol. 7, no. 2, pp. 179–188, Sep. 1936.

[22] D. Li, L. Luo, W. Zhang, F. Liu, and F. Luo, "A genetic algorithm-based weighted ensemble method for predicting transposon-derived piRNAs," *BMC Bioinf.*, vol. 17, no. 1, Dec. 2016, Art. no. 329.

[23] D. Ravì, C. Wong, F. Deligianni, M. Berthelot, J. Andreu-Perez, B. Lo, and G.-Z. Yang, "Deep learning for health informatics," *IEEE J. Biomed. Health Informat.*, vol. 21, no. 1, pp. 4–21, Jan. 2017.

[24] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[25] S. Khan, M. Khan, N. Iqbal, T. Hussain, S. A. Khan, and K.-C. Chou, "A two-level computation model based on deep learning algorithm for identification of piRNA and their functions via Chou's 5-steps rule," *Int. J. Peptide Res. Therapeutics*, vol. 26, no. 2, pp. 795–809, Jun. 2020.

[26] K. Chen and Q. Huo, "Scalable training of deep learning machines by incremental block training with intra-block parallel optimization and blockwise model-update filtering," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2016, pp. 5880–5884.

[27] A. L. Maas, A. Y. Hannun, C. T. Lengerich, P. Qi, D. Jurafsky, and A. Y. Ng, "Increasing deep neural network acoustic model size for large vocabulary continuous speech recognition" 2014, *arXiv:1406.7806*. [Online]. Available: https://arxiv.org/abs/1406.7806

[28] N. Strom, "Scalable distributed DNN training using commodity GPU cloud computing," in *Proc. Annu. Conf. Int. Speech Commun. Assoc. (INTERSPEECH)*, 2015, pp. 1–5.

[29] P. Sinthong, K. Mahadik, S. Sarkhel, and S. Mitra, "Scaling DNN-based video analysis by coarse-grained and fine-grained parallelism," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jul. 2020, pp. 1–6.

[30] T. White, *Hadoop: The Definitive Guide*, vol. 1, 1st ed. Sebastopol, CA, USA: O'Reilly Media, 2009.

[31] M. Khan, "Hadoop performance modeling and job optimization for big data analytics," Ph.D. dissertation, Brunel Univ., London, U.K., 2015.

[32] J. Cózar, F. Marcelloni, J. A. Gámez, and L. de la Ossa, "Building efficient fuzzy regression trees for large scale and high dimensional problems," *J. Big Data*, vol. 5, no. 1, p. 49, Dec. 2018.

[33] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *Proc. USENIX Conf. Hot Topics Cloud Comput.*, 2010, pp. 1–7.

[34] Y. Guo, L. Yu, Z. Wen, and M. Li, "Using support vector machine combined with auto covariance to predict protein–protein interactions from protein sequences," *Nucleic Acids Res.*, vol. 36, no. 9, pp. 3025–3030, May 2008.

[35] Q. Dong, S. Zhou, and J. Guan, "A new taxonomy-based protein fold recognition approach based on autocross-covariance transformation," *Bioinformatics*, vol. 25, no. 20, pp. 2655–2662, Oct. 2009.

[36] S. Chakraverty, D. M. Sahoo, N. R. Mahato, S. Chakraverty, D. M. Sahoo, and N. R. Mahato, "Perceptron learning rule," in *Concepts of Soft Computing*. Singapore: Springer, 2019.

[37] C.-I. Chen and G. W. Chang, "A two-stage ADALINE for harmonics and interharmonics measurement," in *Proc. 5th IEEE Conf. Ind. Electron. Appl.*, Jun. 2010, pp. 340–345.

[38] G. Hinton, L. Deng, D. Yu, G. Dahl, A.-R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Nov. 2012.

[39] T. N. Sainath, A.-R. Mohamed, B. Kingsbury, and B. Ramabhadran, "Deep convolutional neural networks for LVCSR," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 8614–8618.

[40] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. 25th Int. Conf. Neural Inf. Process. Syst.*, vol. 1, 2012, pp. 1097–1105.

[41] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1915–1929, Aug. 2013.

[42] J. Tompson, A. Jain, Y. LeCun, and C. Bregler, "Joint training of a convolutional network and a graphical model for human pose estimation," Jun. 2014, *arXiv:1406.2984*. [Online]. Available: http://arxiv.org/abs/1406.2984

[43] T. Mikolov, S. Kombrink, L. Burget, J. Cernocky, and S. Khudanpur, "Extensions of recurrent neural network language model," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2011, pp. 5528–5531.

[44] K. Wang, J. Hoeksema, and C. Liang, "PiRNN: Deep learning algorithm for piRNA prediction," *PeerJ*, vol. 6, p. e5429, Aug. 2018.

[45] Y. Zuo, Q. Zou, J. Lin, M. Jiang, and X. Liu, "2lpiRNApred: A two-layered integrated algorithm for identifying piRNAs and their functions based on LFE-GM feature selection," *RNA Biol.*, vol. 17, no. 6, pp. 892–902, Jun. 2020.

[46] S. Khan, M. Khan, N. Iqbal, S. A. Khan, and K.-C. Chou, "Prediction of piRNAs and their function based on discriminative intelligent model using hybrid features into Chou's PseKNC," *Chemometric Intell. Lab. Syst.*, vol. 203, Aug. 2020, Art. no. 104056.

[47] J. Ma, R. P. Sheridan, A. Liaw, G. E. Dahl, and V. Svetnik, "Deep neural nets as a method for quantitative structure-activity relationships," *J. Chem. Inf. Model.*, vol. 55, no. 2, pp. 263–274, 2015.

[48] M. K. K. Leung, H. Y. Xiong, L. J. Lee, and B. J. Frey, "Deep learning of the tissue-regulated splicing code," *Bioinformatics*, vol. 30, no. 12, pp. 121–129, 2014.

[49] M. Helmstaedter, K. L. Briggman, S. C. Turaga, V. Jain, H. S. Seung, and W. Denk, "Connectomic reconstruction of the inner plexiform layer in the mouse retina," *Nature*, vol. 500, p. 168, Aug. 2013.

[50] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *J. Mach. Learn. Res.*, vol. 9, pp. 249–256, May 2010.

[51] B. Karlik and A. Olgac, "Performance analysis of various activation functions in generalized MLP architectures of neural networks," *Int. J. Artif. Intell. Expert Syst.*, vol. 1, no. 4, pp. 111–122, 2011.

[52] K. Fredenslund. *Computational Complexity of Neural Networks*. Accessed: Jun. 19, 2020. [Online]. Available: https://kasperfred.com/series/introduction-to-neural-networks/computational-complexity-of-neural-networks

[53] B. Liu, H. Wu, and K.-C. Chou, "Pse-in-one 2.0: An improved package of Web servers for generating various modes of pseudo components of DNA, RNA, and protein sequences," *Natural Sci.*, vol. 9, no. 4, pp. 67–91, 2017.

[54] B. Liu, F. Liu, X. Wang, J. Chen, L. Fang, and K.-C. Chou, "Pse-in-one: A Web server for generating various modes of pseudo components of DNA, RNA, and protein sequences," *Nucleic Acids Res.*, vol. 43, no. W1, pp. W65–W71, Jul. 2015.

[55] Z. Chen, P. Zhao, F. Li, T. T. Marquez-Lago, A. Leier, J. Revote, Y. Zhu, D. R. Powell, T. Akutsu, G. I. Webb, K.-C. Chou, A. I. Smith, R. J. Daly, J. Li, and J. Song, "iLearn: An integrated platform and meta-learner for feature engineering, machine-learning analysis and modeling of DNA, RNA and protein sequence data," *Briefings Bioinf.*, vol. 21, no. 3, pp. 1047–1057, 2020.

[56] R. Guo, Y. Zhao, Q. Zou, X. Fang, and S. Peng, "Bioinformatics applications on apache spark," *GigaScience*, vol. 7, no. 8, pp. 1–10, Aug. 2018.

[57] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauly, M. J. Franklin, S. Shenker, and I. Stoica "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *Proc. 9th USENIX Conf. Netw. Syst. Des. Implement. (NSDI)*, 2012, pp. 15–28.

[58] H. Karau and R. Warren, *High Performance Spark: Best Practices for Scaling and Optimizing Apache Spark*, 1st ed. Sebastopol, CA, USA: O'Reilly Media, 2017.

[59] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Trans. Autom. Control*, vol. 31, no. 9, pp. 803–812, Sep. 1986.

[60] J. Chen, R. Monga, S. Bengio, and R. Józefowicz, "Revisiting distributed synchronous SGD," *CoRR*, 2016.

[61] H. Cui, H. Zhang, G. R. Ganger, P. B. Gibbons, and E. P. Xing, "GeePS: Scalable deep learning on distributed GPUs with a GPU-specialized parameter server," in *Proc. 11th Eur. Conf. Comput. Syst. (EuroSys)*, 2016, pp. 1–16.

[62] M. F. Sabooh, N. Iqbal, M. Khan, M. Khan, and H. F. Maqbool, "Identifying 5-methylcytosine sites in RNA sequence using composite encoding feature into Chou's PseKNC," *J. Theor. Biol.*, vol. 452, pp. 1–9, Sep. 2018.

[63] H. Byun and S. W. Lee, "Applications of support vector machines for pattern recognition: A survey," in *Pattern Recognition With Support Vector Machines*. Berlin, Germany: Springer, 2002, pp. 213–236.
[64] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, "KNN model-based approach in classification," in *Proc. OTM Confederated Int. Conf. Move Meaningful Internet Syst.*, in Lecture Notes in Computer Science: Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics, 2003, pp. 213–236.
[65] V. Svetnik, A. Liaw, C. Tong, J. C. Culberson, R. P. Sheridan, and B. P. Feuston, "Random forest: A classification and regression tool for compound classification and QSAR modeling," *J. Chem. Inf. Comput. Sci.*, vol. 43, no. 6, pp. 1947–1958, Nov. 2003.
[66] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in *Proc. Spring Joint Comput. Conf. (AFIPS Spring)*, Apr. 1967, pp. 483–485.
[67] M. Khan, P. M. Ashton, M. Li, G. A. Taylor, I. Pisica, and J. Liu, "Parallel detrended fluctuation analysis for fast event detection on massive PMU data," *IEEE Trans. Smart Grid*, vol. 6, no. 1, pp. 360–368, Jan. 2015.
[68] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Proc. IEEE Eur. Symp. Secur. Privacy (EuroS&P)*, Mar. 2016, pp. 372–387
[69] C. Ferreira, "Gene expression programming: A new adaptive algorithm for solving problems," *Complex Syst.*, vol. 13, no. 2, pp. 87–129, 2001.
[70] J. Kennedy, R. C. Eberhart, and Y. Shi, *Swarm Intelligence*. Amsterdam, The Netherlands: Elsevier, 2001.

SALMAN KHAN received the M.S. degree in mobile computing from the University of Glamorgan, U.K., in 2009. He is currently pursuing the Ph.D. degree with the Department of Computer Science, Abdul Wali Khan University Mardan. He is also a Lecturer with the Department of Computer Science, Abdul Wali Khan University Mardan. He has about nine year experience of teaching and industry together. His areas of interest are big data, mobile computing, bioinformatics, machine learning, and parallel programming.

MUKHTAJ KHAN received the Ph.D. degree in performance modeling and big data analytics from the Department of Electronics and Computer Engineering, Brunel University London, U.K., in 2015. He was a Postdoctoral Fellow with the School of Engineering, Design and Physical Sciences, Brunel University London, in 2015 and 2016. He is currently working as an Assistant Professor with the Department of Computer Science, Abdul Wali Khan University Mardan, Pakistan. He has authored over 30 research articles published in journals and conference proceedings. His research interests include performance modeling, big data analytics, parallel computing, and machine learning.

NADEEM IQBAL (Senior Member, IEEE) received the Ph.D. degree in bio and brain engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2013. He was a Postdoctoral Fellow with the School of Mechanical Engineering, University of Leeds, U.K. He is currently working as an Assistant Professor with the Department of Computer Science, Abdul Wali Khan University Mardan, Pakistan. His research interests include supervised and unsupervised machine learning techniques for control prosthesis, biological information processing mechanism in brain, and pattern recognition.

MAOZHEN LI received the Ph.D. degree from the Institute of Software, Chinese Academy of Sciences, in 1997. He was a Postdoctoral Research Fellow with the School of Computer Science and Informatics, Cardiff University, U.K., from 1999 to 2002. He is currently a Professor with the Department of Electronic and Computer Engineering, Brunel University London, U.K. His research interests include the areas of high performance computing (grid and cloud computing), big data analytics, and intelligent systems. He is on the editorial boards of *Computing and Informatics* Journal and the *Journal of Cloud Computing: Advances, Systems and Applications*. He has more than 100 research publications in these areas. He is a Fellow of the British Computer Society.

DOST MUHAMMAD KHAN (Member, IEEE) received the bachelor's degree in statistics and the master's and Ph.D. degrees in statistics from the University of Peshawar, Pakistan, in 2000, 2003, and 2012, respectively. He was also a Visiting Research Fellow with the School of Statistics, University of Minnesota, USA, in 2008. He is currently working as an Assistant Professor with the Department of statistics, Abdul Wali Khan University Mardan, Pakistan. His research interests include robust statistics, applied statistics, mathematical statistics, and computational statistics.