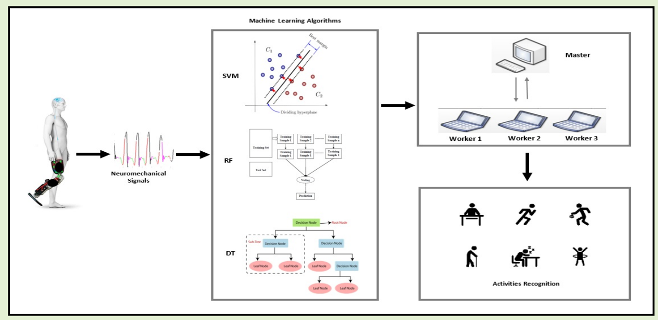
Neuromechanical Signal-Based Parallel and Scalable Model for Lower Limb Movement Recognition

Nadeem Iqbal, *Senior Member, IEEE*, Tufail Khan, Mukhtaj Khan, Tahir Hussain, Tahir Hameed, and Syed Ahmad Chan Bukhari, *Senior Member, IEEE*

***Abstract*—Individual who have lost their lower limb because of amputation can use the prosthesis to restore daily living activities. The amputee intent recognition during locomotion modes can be used as source to control lower limb prosthesis. Due to continuous data recording from multiple sensors, the timely recognition of activities of daily living have become a challenging issue for traditional technology and conventional machine learning algorithms. This work hypothesize that parallel discriminant features can be learned from large amount of data generated by aggregating the neuromechanical signals from multiple subjects with parallel and distributed computing platform. Consequently, this paper apply three classifiers including support vector machine, decision tree and random forest on large data sets. The model performance is extensively evaluated in terms of different performance measurement parameters such as accuracy, efficiency, scalability and speedup in sequential and distributed environment. The experimental results show that the parallel approach achieved 3.9x computation speedup as compared to the sequential approach without affecting accuracy level. The parallel support vector machine algorithm demonstrated high speedup and scalability in comparison with random forest and decision tree algorithms. The outcome of this study could promote parallel based model for the unobtrusive recognition of lower limb locomotion modes and could promote the future design for the intelligent control of prostheses and exoskeleton.**



***Index Terms*— Bilateral sensors fusion, support vector machine, IMU, neuromechanical sensors, sEMG.**

I. Introduction

LOWER limb amputation is a major cause of disability, which has severely impacted the physical and physiological abilities of the effected individuals. Artificial intelligence and lower limb robotics can be leveraged to increase the independence of such people and restore their locomotion. These robotics assist the users with versatile function beyond the simple walking activity. However, without prior knowledge of the user's intent for a particular mode of movement, the assistive robotic device cannot properly select the mode of movement or adjust the joint motion [1]. The mode of motion in the current assistive robotics is changed manually which is cumbersome and does not allow natural transition between different motion modes such as level ground walk, ramp ascent/descent and stair ascent/descent. Therefore, the development of an interface for intelligent locomotion mode recognition has been drawing increased attention [2].

Surface electromyography (sEMG) of the lower limb is one of the major techniques for intent recognition and intelligent control of lower limb robotics. sEMG directly provides the user intent for a particular mode of movement and has been proposed as a control signal for lower limb robotics [3]. However, sEMG signals are noisy, non-stable and may be difficult to use independently for controlling an assistive robotics such as lower limb prosthesis during different locomotion modes. Consequently, a complementary approach uses multimodal information i.e. to fuse sEMG with the inertial measurement unit (IMU) sensors. The fusion of sEMG and IMU sensors offers improved system performance as well as improved robustness. The sEMG signals appear prior to user movement and may be used to predict transitions between different tasks

Manuscript received April 1, 2021; accepted April 12, 2021. Date of publication April 27, 2021; date of current version July 14, 2021. The associate editor coordinating the review of this article and approving it for publication was Dr. Varun Bajaj. *(Corresponding author: Nadeem Iqbal.)*

Nadeem Iqbal, Tufail Khan, Mukhtaj Khan, and Tahir Hussain are with the Department of Computer Science, Abdul Wali Khan University, Mardan 23200, Pakistan (e-mail: nikhan@awkum.edu.pk; tufailshah.cs@gmail.com; mukhtaj.khan@awkum.edu.pk; tahirhussain1983@yahoo.com).

Tahir Hameed is with Merrimack College, North Andover, MA 01845 USA (e-mail: hameedt@merrimack.edu).

Syed Ahmad Chan Bukhari is with the Division of Computer Science, Mathematics and Science, Collins College of Professional Studies, St. John's University, New York, NY 11439 USA (e-mail: bukharis@stjohns.edu).

Digital Object Identifier 10.1109/JSEN.2021.3076114

whereas IMU sensors respond to the user's movements [4]–[6]. Artificial intelligence can be leveraged here as a bridge between multiple signals and the robotic behavior [7]–[9].

One of the main factors behind accurate predictions is the availability of large amounts of training data. Lower limb activities involve quite dynamic and complex motions where a single subject cannot practically be expected to record thousands of motion datapoints/examples in a single session. Therefore, in order to obtain large amounts of training data, the recordings from multiple subjects can be aggregated, thus fostering the environment necessary for learning about a general mapping of the subject's movements. This mapping in turn assists the locomotion modes recognition of new subjects [10], [11].

In this paper, we propose a systematic approach towards recognizing lower limb locomotion modes using three machine learning algorithms. In domains such as human activity recognition (HAR), object recognition and natural language processing (NLP), machine learning performance has been very profound. Because of the great importance and learning capability of systems using machine learning, this study employed three algorithms in sequential and a parallel environment using Apache Spark framework. Accuracy and efficiency of the model were cross-checked with parallel versions of the traditional machine learning algorithms including Support Vector Machine, Decision Tree and Random Forest. This study aims to extend previous research as follows.

- Propose a parallel, scalable, fast and fault-tolerant model for the activities of daily livings based on bilateral neuromechanical signal
- The proposed model used both IMU and sEMG sensor's data on number of processing worker nodes using spark framework to achieve parallelism with high activities recognition.
- The performance of the classifiers are extensively evaluated using different performance measurement metrics such as accuracy, execution time, speedup and scalability.

The proposed parallel activity recognition model can significantly reduce the computational time and the required amount of memory. We employed spark framework to compare the results of sequential and parallel machine learning algorithms i.e. Support Vector Machine (SVM), Decision Tree (DT) and Random Forest (RF). In addition, the application of bilateral neuromechanical activity dataset verify that our proposed method performs well for big datasets where the activity recognition has millions of observations.

The rest of the paper is organised as follows: Section II describes related work. Section III illustrates the proposed method in detail. The results and discussion are presented in Section IV and V respectively. Finally, the conclusion and future work of this work is presented in Section VI.

II. Related Work

Recently, activity recognition has received increasing attention within the field of human healthcare and lower limb prosthetics due to their success in other domains such as recognition of objects and faces. A sequential Linear Discriminant Analysis (LDA) classifier was used by the authors in [12] to classify five activities of a lower limb amputee. Thirteen mechanical sensors placed on different positions of a powered prosthetic limb were used in this study. Although accuracy of the system was better for transitional activities, but the use of thirteen sensors made the system complex. Furthermore, the activities tested were few in number (only five). The researchers in [13] developed an activity recognition system using sensors such as triaxial gyroscope, a triaxial accelerometer, and a magnetometer. Five sequential classification algorithms including SVM, Dynamic Time Warping (DTW), Artificial Neural Network (ANN), K-nearest neighbor (KNN) and Bayesian Decision Making (BDM) were used. For training and testing purposes, different cross-validation techniques were used, such as P-fold etc. It was concluded that sequential BDM performed better than other classifiers. However, for certain activities such as stair ascending and descending, it achieved low accuracy (52.3%–79.4%).

Myoelectric signals such as electromyography (EMG) is increasingly being used for the classification of different human activities nowadays. The technique through which electrical signals produced by different body muscles are recorded and processed is known as EMG. The authors in [14] recorded EMG signals from fifteen subjects for the classification of eight activities. An accuracy of 81% was obtained after combining sixteen domain features.

Sequential machine learning algorithms such as deep neural network need more data for training and they also require huge amount of computational resources. This involves longer training times which can range from several hours to days even with general purpose graphics processing unit (GPGPU) based acceleration [15], [16]. In order to increase the efficiency of such algorithms, several approaches have been proposed. One approach is to significantly reduce training time with the use of highly optimized General-Purpose Graphics Processing Units (GPGPU), often showing 10–100×speed-up [15]. However, this approach also suffers from some limitations. A single machine has limited resources such as GPU and main memory, so speeding up model training on one machine is difficult to be achieved [17]. The authors in [18], [19] and [20] suggested scaling out methods in distributed environments to deal with such issues. These approaches use model parallelism and can possibly provide scalability. To accelerate model training, the authors in [21] proposed DeepSpark which is a new deep learning framework on Spark, and which addresses the issues faced in large-scale data handling. The authors in [22] developed BigDL, a distributed framework for big data platforms and workflows. BigDL supports an API similar to Torch and Keras for constructing neural network models. It also supports both large-scale distributed training and inference. The authors in [23] proposed an approach to perform Mobile Big Data (MBD) analytics using Deep Learning and Apache Spark Framework. In a high-performance computing cluster, the authors parallelized the training and learning of deep learning models, with the intention that their proposed framework would speed up Mobile Big Data decision-making process. However, the biggest drawbacks of deep learning algorithms is the high computation cost, inter-processor communication bottlenecks and parameters training time.

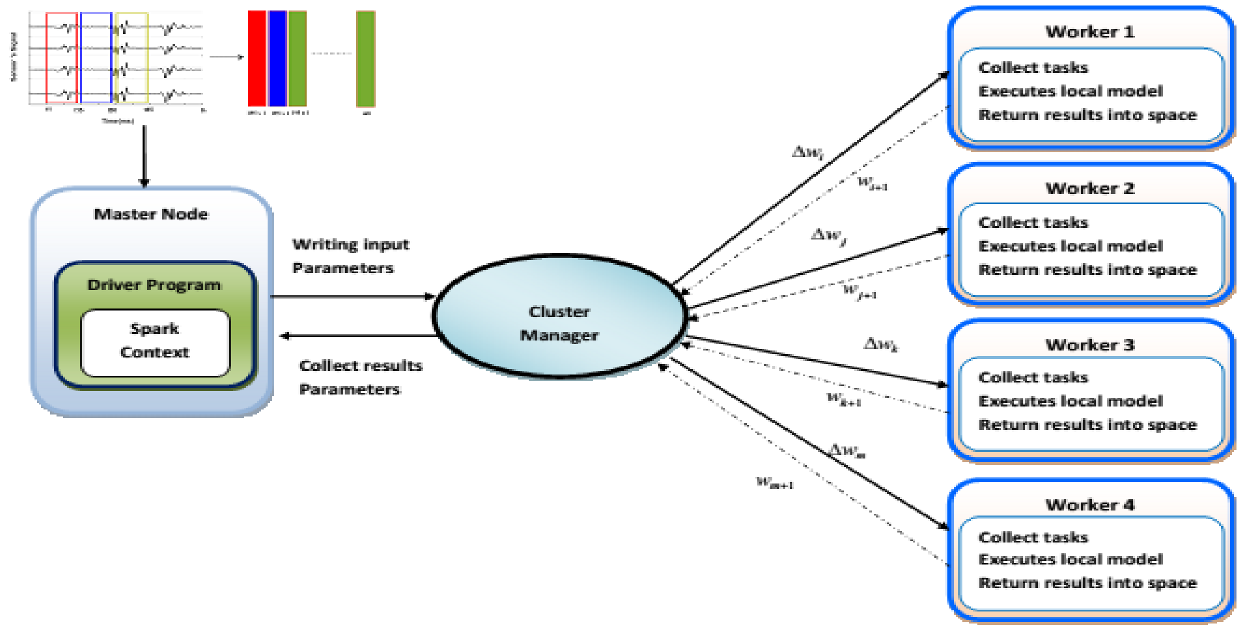


Fig. 1. Flow Diagram of the Spark framework.

There has been a lot of research on human activity recognition using deep learning but, when the amount of data is large, the computation of human activity recognition data using conventional computing methods is very slow and the computation is a bottleneck for application of activity recognition on massive datasets. This study used Apache Spark to parallelize lower limb activity recognition on three machine learning algorithms i.e. DT, SVM, and RF. Results show that the parallel machine learning approach achieved 3.9x speedup as compared to the sequential approach using four physical processing nodes. Moreover, the system is more scalable, accurate and fault tolerant as compared to the sequential machine learning approach.

III. Data and Methods

A. Dataset

In this study we have utilized a publically available dataset [24] to train and test our model. This dataset comprises recordings of 10 able-bodied subjects (7 male, 3 female; age: 23–29 years). The subjects were instrumented bilaterally with wearable IMU, Goniometer and sEMG sensors to measure lower limb muscle activity and limb kinematics. We only considered IMU and sEMG sensors. sEMG signals were recorded using bipolar surface electrodes (DE2.1, Delsys, Boston, MA, USA) from the same seven muscles in each leg. The targeted muscles were biceps femoris (BF), medial gastrocnemius (MG), soleus (SOL), vastus lateralis (VL), tibialis anterior (TA), rectus femoris (RF), and semitendinosus (ST). These muscles are responsible for hip and knee flexion/extension and ankle plantar flexion/dorsiflexion, movements that are commonly assisted by wearable devices. Signals from the sEMG sensors were collected at 1 kHz. 6-DOF IMU's (tri-axial accelerometer and tri-axial gyroscope) were placed on the subject's thigh and shank and sample at 500Hz. Each subject performed seven steady state activities consisting of standing (St), sitting (S), level ground walking (LW), ramp ascending/descending (RA/RD), stair ascending/descending (SA/SD) and seven transitional activities including S $\rightarrow$ St $\rightarrow$ LW $\rightarrow$ RA $\rightarrow$ LW $\rightarrow$ SD $\rightarrow$ LW $\rightarrow$ St. These activities were chosen because they include the most common types of terrains likely encountered in community ambulation.

B. Preprocessing

The raw sensors readings of sEMG and IMU sensors were stacked to create batches of data. The raw signals form all the 13 channels (triaxial accelerometer, triaxial gyroscope and seven sEMG channels) were segmented into 150ms (0.15s) time windows, which resulted into 15×13 matrix. We stacked the signals all the three types of sensors which results into 195 by 1 matrix. For example, for the sample time t, we stacked the signal from all 13 sensors from $t - 15$ until t into a matrix of size 195 by 1. This matrix corresponds to 1 sample and this process is repeated 500 times. Fig. 1 shows that the given raw signal is segmented into multiple windows and given to the Spark framework.

C. Experimental Setup

The proposed cluster was comprised of four (04) Intel Core i3 computers, each having 4 CPUs, 8 GB RAM and a 500 Gigabyte (GB) hard disk storage. Apache Spark 2.4.0 was installed on each physical processing node. Python programming language was used for writing all programming codes. All the nodes were equipped with Anaconda version 3.0 along with Jupyter Notebook. One processing node was configured as a namenode or server machine. The server machine along with the remaining three other machines were configured as Data Nodes.

D. Spark Architecture

Apache Spark is a high-level and general-purpose cluster computing platform designed to be fast and fault-tolerant. Apache Spark is an open source big data processing framework [25]. Spark is a memory-efficient computation framework where the data is maintained and processed in shared physical memory. The Spark is an efficient computational framework for iterative machine learning algorithms and supports multiple programming languages such as Python, Java, R and Scala. The Spark architecture is comprised of driver program, cluster manager Spark Context and worker node as shown in Fig 1. Apache Spark's architecture has Master-Slave framework. In this architecture there is one main/central coordinator and there are many distributed worker nodes. The main/central coordinator is known as Spark Driver. Communications with all the workers is Spark driver's responsibility. One or more executors are running on every worker node. Executors' job is to process the task. Executors have to register themselves with the Driver. The Driver has all the information about Executors. A spark application is actually the working combination of Driver and Workers.

A number of built-in libraries are included in Spark such as SQL, GraphX, MLib, and Spark Streaming to facilitate the development of applications in various domains. In addition to the in-built libraries, an important memory abstraction introduced in the Spark framework is the resilient distributed dataset (RDD), which is fault tolerant and provides scalability. In order to achieve fault tolerance, Apache Spark distributes several copies of RDDs across multiple processing nodes. RDD lineage is used to achieve fault tolerance. When a machine holding RDD data fails (machine failure is a common phenomenon in computing), Apache Spark, without letting the users know about this failure, uses this ability and recompute the missing/lost partitions. The role of fault-tolerance is that model doesn't stop working when a partition is lost as it can always be recomputed.

E. Machine Learning Algorithms

Three supervised learning algorithms were used to build the activity recognition models: SVM, DT and RF. SVM can handle high dimensional data well and produce a compact model. DTs are good for achieving high activity recognition accuracy and RF combine the predictive capabilities of multiple activity learners.

SVM: Support vector machine is a classification algorithm used to classify signals, images, body movements etc. SVM performs classification by constructing a separating hyperplane that differentiates between classes [26]. For nonlinear separable problems the SVM maps the input space into a high dimensional feature space using nonlinear function called kernels and constructs an optimal hyperplane. For nonlinear input training data $(\mathrm{x}_1, \mathrm{y}_1) \ldots (\mathrm{x}_\mathrm{m}, \mathrm{y}_\mathrm{m)} \in \mathrm{R}^\mathrm{N}\mathrm{x}\{-1, +1\}$ where x_i is the input value and y_l shows the assigned class to which it belongs, SVM maps these data into a new feature space R^M using the nonlinear transformation φ: RN $\rightarrow$ RM. The hyperplane can be represented with the equation

$$\omega \, . \, \varphi(x) + b = 0 \tag{1}$$

where $\omega \in \mathrm{R}^M$ and $b \in \mathrm{R}$

The hyperplane determines the margin between the classes. The data points for each class are located on one side of the hyperplane. For each class, the distance from the closet point the hyperplane is equal [2].

Decision Tree: A decision tree is a supervised classification algorithm that contains conditional control statements. A decision tree is represented in a flowchart like structure where each internal node represents a test on an attribute, each branch shows the output of the test and each leaf node is a class label. To reach the best possible performance, the design of the decision tree is of utmost importance [27]. The choice of tree structure and the choice of appropriate feature subsets will be reflected in classification accuracy and efficiency. To predict responses to data, the decision tree follows the decisions from the root node to the leaf node. The leaf nodes contain the response of the data.

A decision tree is constructed from the pre-classified data. The division of data into different classes is based on the values of the features of the given data. This division process is applied to all subsets of data items recursively. The process terminates as for as all the data items in the current subset belong to the same class.

Random Forest: Random forest is a supervised learning algorithm which is an ensemble of regression and classification trees. RF creates a set of decision trees using random resampling on the training set. It is superior than a single decision tree because it eliminates over-fitting by averaging out the result. Once the tree is built, a set of bootstraps, which is different from the original training dataset out-of-bag (OOB) samples, is used as the testing set [6]. The error rate of the classification of all the testing sets is the OOB estimate. Thus, using the OOB estimate removes the need for cross-validation or a separate test set. To classify new input data, each individual RF tree votes for one class and the forest predicts the class using majority voting [28].

F. Learning Algorithms Parallelization

In this section, we introduce parallel machine learning algorithms using Spark framework. In the Spark architecture the large training data are divided into samples of RDDs. These RDDs are then distributed across multiple nodes of the Spark cluster. Distributing the RDDs among the multiple nodes attempt to achieve a high data level parallelization. Moreover, the Spark framework is automatically shared/replicated a copy of each classification model (DT, SVM and RF) across multiple workers nodes of the cluster to achieve model level parallelization. The training process is started by executing the model across the worker nodes in parallel. Local model's parameters are reported by the worker node to the namenode (parameter server). In order to get a global model, the local model's parameters are aggregated by the parameter server and their average/mean is calculated. Once this process is completed, the parameters are updated by the parameter server and these new parameters are shared with the worker nodes. The last three steps i.e. training of the local model by each worker node, aggregation of local parameters by the parameter server to get a global model and updating and sharing of new parameters by the parameter server with

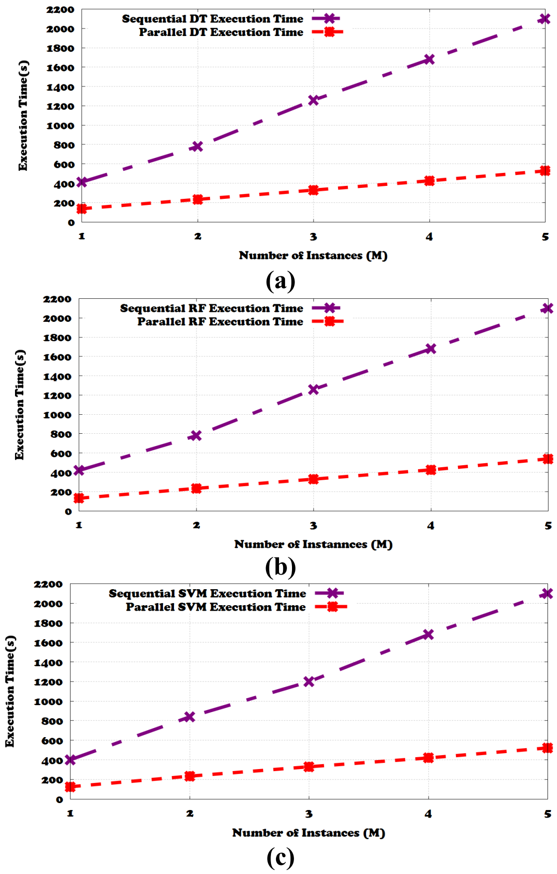


Fig. 2. Efficiency comparison of parallel DT,SVM and RF with sequential DT,SVM and RF for different number of signal instances.

every worker, are repeated multiple times in order to get a global trained model with optimum values of the model hyperparameters.

IV. Results

A. Parallel Models' Efficiency Comparison With Sequential Models

This section provides a comparison of the proposed parallel models with sequential approach. The execution time of parallel approach was compared with the execution/running time of sequential approach using different number of instances (one, two, three four and five million). Fig. 2 (a, b, c) illustrates the experimental results of DT, SVM and RF for sequential and parallel execution respectively. It is evident that the execution/running time of sequential approach is drastically increased when we increase the number of signal instances.

As compared to sequential approach, the execution/running time of parallel approach is significantly decreased while increasing number of signal instances. In order to further process four million signal instances, the sequential Decision Tree's execution time was 1680 seconds whereas the parallel Decision Trees execution time was 458 seconds using four processing nodes. From the results, it can be concluded that the parallel DT reduced the execution time 3.98x times as compared to the sequential DT.

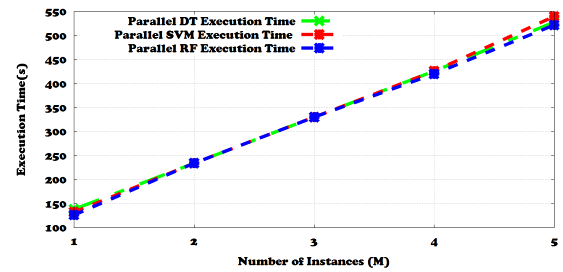


Fig. 3. Parallel models' performance.

B. Model Performance in Terms of Running Time

Fig. 3 demonstrates parallel models' performance in term of their running time. A gradual increase in the execution time of these algorithms was observed while processing one million and five million instances respectively. For the remaining cases, there was no variation and the execution time for parallel RF, SVM and DT remains the same. For example, while processing one million signal instances, the execution time of parallel DT, parallel RF and parallel SVM is 138s, 132s and 126s respectively. Likewise, with five million instances, the running/execution time of parallel DT, parallel RF and parallel SVM is 528s, 540s and 522s respectively.

C. Scalability Analysis of the Proposed Parallel Model

On a parallel architecture, the scalability of a parallel algorithm is a measure of its capacity to effectively utilize an increasing number of processors. This section describes the scalability analysis of the parallel models using one, two, three and four processing nodes along with different number of signal instances. Fig. 4 presents the scalability analyses of the parallel DT, SVM and RF models. It is obvious that, a continuous decreasing trend is observed in execution time of the parallel algorithm when the number of processing nodes for all cases of signal instances are increased. For the four million instances, the parallel DT's execution time was 1680 seconds using a single processing node while its parallel version took 426 seconds using 4 processing nodes.

This confirms that increasing the number of physical processing nodes significantly reduces algorithm's execution time.

D. Accuracy Analysis of the Proposed Parallel Model

The most important and critical performance measurement parameter for the evaluation of a classification model is accuracy. In simplest words, accuracy in classification is the ratio of correctly predicted observations to the total observations. Being one of the most important performance measurement parameters, accuracy of the parallel approaches in comparison with the sequential approaches have extensively been evaluated.

Table I illustrate the accuracy comparison of the proposed model. The highest average accuracy was achieved by DT whereas the lowest accuracy was observed in the case of SVM. For sequential approach, DT achieved 99.5% accuracy (Acc) 96.4% sensitivity (Sns), 99.3% specificity (Spc) and 97.2% Mathew's coefficient of correlation (Mcc) respectively.

TABLE I
ACCURACY ANALYSIS OF THE PROPOSED MODEL

	DT				SVM				RF			
Activity	Acc	Sns	Spc	Mcc	Acc	Sns	Spc	Mcc	Acc	Sns	Spc	Mcc
1	100.0	98.6	99.7	99.1	85.3	0	100.0	0	99.6	100.0	100.0	98.4
2	98.0	90.2	98.7	92.4	61.4	21.1	61.2	16.8	99.6	97.7	99.5	98.6
3	100.0	99.0	99.0	98.9	86.7	53.1	88.5	56.3	99.6	97.5	99.6	98.5
4	100.0	98.7	99.8	97.9	86.5	69.6	87.6	54.4	99.2	97.6	99.6	96.3
5	99.9	97.4	99.5	97.6	88.7	64.5	93.2	57.8	99.3	100.0	100.0	97.6
6	99.0	91.2	98.5	94.8	65.7	21.9	67.1	17.3	98.5	96.6	99.4	98.0
7	100.0	100.0	100.0	100.0	98.5	95.3	99.1	94.4	98.8	99.0	0	99.4
Average	99.5	96.4	99.3	97.2	81.8	46.5	85.2	42.4	98.5	98.3	85.4	98.1

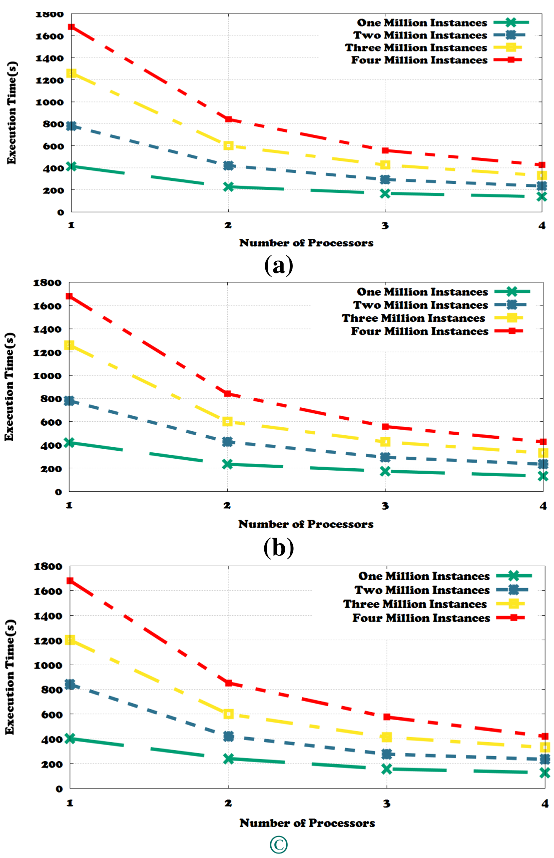


Fig. 4. a) DT b) RF and c) SVM's scalability using different number (1, 2, 3, 4) of processing nodes.

Likewise, SVM achieved 81.8%Acc, 46.5% Sns, 85.2%Spc, and 42.4%Mcc. Similarly, RF obtained 98.5% Acc, 98.3% Sns, 85.4% Spc, and 98.1%Mcc.

E. Speedup Analysis of the Proposed Parallel Model

Speedup is the ratio of the time taken to solve a problem on a single processor to the time required to solve the same problem on a parallel system with 'P' identical processing elements. The speedup of parallel algorithms performance has been analyzed in this section. Mathematically, the speedup is computed as

$$SP = \frac{T_1}{T_n} \tag{2}$$

where SP represents the relative speedup of training model with n worker nodes, T_1 is the execution time of one processing node whereas T_n represents the execution time using n number of processing nodes ($n = 4$). The speedup for parallel DT, SVM and RF is presented in Fig. 5. The results exhibit that for four processing nodes, the parallel algorithms achieved more than 3x times speedup.

The Parallel efficiency of the proposed parallel model can be computed further using eq. (3)

$$EP = \frac{SP}{n} \tag{3}$$

where EP is the parallel efficiency on the Spark cluster with n worker nodes which shows the utilization degree of all computing resources. Fig 5 shows the parallel efficiency with one, two, three, four and five million instances with different number of worker nodes. It is evident that the average parallel efficiency for DT, SVM and RF reaches up to 98% which shows that all the computing resources of the Spark are efficiently utilized. As can be seen in Fig. 5 when size of dataset is increased the obtained parallel efficiency is increased. For example, when the number of worker nodes is three the parallel efficiency is more than 98% for five million instances. Therefore, the proposed parallel activity recognition method can make full use of the computing resources of a Spark to efficiently process large-scale neuromechanical signals in parallel.

V. DISCUSSION

Machine learning offers a wide range of statistical algorithms for human activity recognition. It includes various

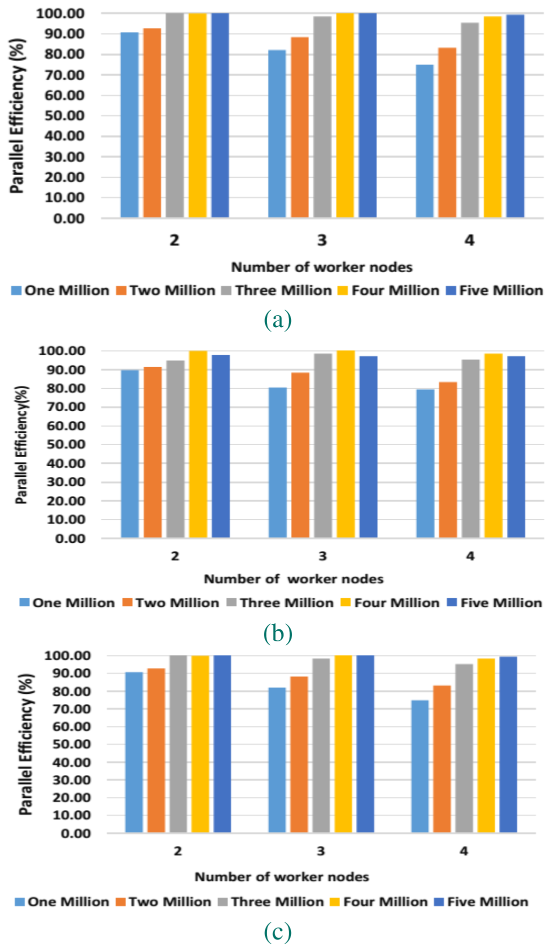


Fig. 5. Parallel efficiency a) DT b) SVM c) RF using different number of signal instances.

techniques such as SVM, DT, RF and other data mining techniques. All these algorithms are computationally expensive on massive datasets which makes them the ideal cases for implementation using parallel architecture. The proposed activity recognition approach parallelized three conventional machine learning algorithms including DT, SVM, and RF using Apache Spark framework having four physical nodes. The performance of the proposed method was examined using sequential and parallel execution models. Sequential machine learning models such as DT, SVM and RF have their own merits and demerits but use of parallel models is on the rise due to their ability to indirectly learn different activity patterns from raw sensor's signals. Parallel learning algorithms are unique due to their ability to accelerate processing which assists in accurately recognizing real-time human activities. In this study, first we parallelized three conventional machine learning algorithms (DT, SVM and RF) and analyzed their efficiency, accuracy, scalability and speed with their sequential counterparts.

In the first study, the execution time for both sequential and parallel activity recognition model was examined. It was observed that increasing the number of instances for neuromechanical signals the execution time of sequential activity recognition approach was significantly high. Whereas, the execution time of parallel models were slightly increased when the number of signals instances were increased.

Running time is important factor for measuring the performance of an activity recognition system [29]. While evaluating the performance of sequential and parallel models, the number of signal's instance were gradually increased. The more the number of instances, the more time is required to run the model. For example, processing three million signal instances, the sequential RF's execution time was 1260 seconds, whereas, the parallel RF's execution time was only 330 seconds. It means that proposed parallel RF decreased the execution time 3.8x times as compared to sequential RF. Likewise, with five million instances, the running/execution time of parallel DT and parallel SVM, were 528 and 522 seconds respectively. A continuous decreasing trend was observed in execution time of the parallel models when the number of processing nodes were increased for all cases of signal instances. For example, in the case of four million instances, the parallel DT's running time was about 1680 seconds using single processing node while its parallel version took 426 seconds using four processing nodes. It is obvious that increasing the number of physical processing nodes considerably reduces model's execution time.

With four processing nodes the parallel activity recognition approach attained more than 2.5x speedup for all instances. For instance, RF has reported maximum speed up of 2.95 on three million signal instances using three processing nodes and 3 speedup on four million signal instances using the same number of processing nodes. On average, the parallel RF obtained more than 3.6x speedup using four physical processing nodes. However, on the average, the parallel algorithms speedup is less than the total number of processing nodes. It's due to the fact that parallel processing involved communication and task initialization overheads which is collectively known as processing overhead.

VI. Conclusion and Future Work

This study proposed fast, scalable and fault-tolerant parallel and distributed multisensor based models for lower limb activity recognition. The proposed approach parallelized three conventional machine learning algorithms including DT, SVM and RF using Apache Spark framework having four physical nodes and computed their efficiency, accuracy, scalability and speedup analysis on raw heterogeneous neuromechanical signals in a parallel and distributed manner. It was demonstrated that parallelization approach using bilateral, raw neuromechanical signals significantly improved the speed and accuracy of activity classification without requiring preprocessing and features extraction steps. Performance of the proposed approach was assessed from four different aspects: efficiency, accuracy, scalability and speedup. The proposed parallel models showed highest performance in terms of accuracy and efficiency. These encouraging results can aid the future design of intelligent lower limb prosthetics and, therefore, improve the mobility of lower limb amputees. The main limitation of this work is that in the data collection process only able bodied subjects were recruited which performed seven steady state and transitional activities. In the future, a parallelization approach will be adopted on more massive activity datasets of lower limb amputees and the classifiers will be enhanced to classify more complex lower limb activities.

References

[1] H. Huang, T. A. Kuiken, and R. D. Lipschutz, "A strategy for identifying locomotion modes using surface electromyography," *IEEE Trans. Biomed. Eng.*, vol. 56, no. 1, pp. 65–73, Jan. 2009, doi: 10.1109/TBME.2008.2003293.

[2] T. Hussain, N. Iqbal, H. F. Maqbool, M. Khan, M. I. Awad, and A. A. Dehghani-Sanij, "Intent based recognition of walking and ramp activities for amputee using sEMG based lower limb prostheses," *Biocybernetics Biomed. Eng.*, vol. 40, no. 3, pp. 1110–1123, Jul. 2020, doi: 10.1016/j.bbe.2020.05.010.

[3] F. Zhang and H. Huang, "Source selection for real-time user intent recognition toward volitional control of artificial legs," *IEEE J. Biomed. Health Informat.*, vol. 17, no. 5, pp. 907–914, Sep. 2013, doi: 10.1109/JBHI.2012.2236563.

[4] T. Hussain, H. F. Maqbool, N. Iqbal, M. Khan, Salman, and A. A. Dehghani-Sanij, "Computational model for the recognition of lower limb movement using wearable gyroscope sensor," *Int. J. Sensor Netw.*, vol. 30, no. 1, pp. 35–45, 2019, doi: 10.1504/IJSNET.2019.099230.

[5] H. F. Maqbool, M. A. B. Husman, M. I. Awad, A. Abouhossein, and A. A. Dehghani-Sanij, "Real-time gait event detection for transfemoral amputees during ramp ascending and descending," in *Proc. 37th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, Aug. 2015, pp. 4785–4788, doi: 10.1109/EMBC.2015.7319464.

[6] J. Lu and K.-Y. Tong, "Robust single accelerometer-based activity recognition using modified recurrence plot," *IEEE Sensors J.*, vol. 19, no. 15, pp. 6317–6324, Aug. 2019, doi: 10.1109/JSEN.2019.2911204.

[7] E. Zheng, L. Wang, K. Wei, and Q. Wang, "A noncontact capacitive sensing system for recognizing locomotion modes of transtibial amputees," *IEEE Trans. Biomed. Eng.*, vol. 61, no. 12, pp. 2911–2920, Dec. 2014, doi: 10.1109/TBME.2014.2334316.

[8] Z. Liu, W. Lin, Y. Geng, and P. Yang, "Intent pattern recognition of lower-limb motion based on mechanical sensors," *IEEE/CAA J. Autom. Sinica*, vol. 4, no. 4, pp. 651–660, Sep. 2017, doi: 10.1109/JAS.2017.7510619.

[9] A. J. Young and L. J. Hargrove, "A classification method for user-independent intent recognition for transfemoral amputees using powered lower limb prostheses," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 24, no. 2, pp. 217–225, Feb. 2016, doi: 10.1109/TNSRE.2015.2412461.

[10] U. Cote-Allard *et al.*, "Deep learning for electromyographic hand gesture signal classification using transfer learning," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 27, no. 4, pp. 760–771, Apr. 2019, doi: 10.1109/TNSRE.2019.2896269.

[11] T.-H. Tan, M. Gochoo, S.-C. Huang, Y.-H. Liu, S.-H. Liu, and Y.-F. Huang, "Multi-resident activity recognition in a smart home using RGB activity image and DCNN," *IEEE Sensors J.*, early access, Aug. 23, 2018, doi: 10.1109/JSEN.2018.2866806.

[12] A. J. Young, T. A. Kuiken, and L. J. Hargrove, "Analysis of using EMG and mechanical sensors to enhance intent recognition in powered lower limb prostheses," *J. Neural Eng.*, vol. 11, no. 5, Oct. 2014, Art. no. 056021, doi: 10.1088/1741-2560/11/5/056021.

[13] K. Altun, B. Barshan, and O. Tunçel, "Comparative study on classifying human activities with miniature inertial and magnetic sensors," *Pattern Recognit.*, vol. 43, no. 10, pp. 3605–3620, Oct. 2010, doi: 10.1016/j.patcog.2010.04.019.

[14] W. Caesarendra, S. U. Lekson, K. A. Mustaqim, A. R. Winoto, and A. Widyotriatmo, "A classification method of hand EMG signals based on principal component analysis and artificial neural network," in *Proc. Int. Conf. Instrum., Control Automat. (ICA)*, no. 3, Aug. 2016, pp. 22–27, doi: 10.1109/ICA.2016.7811469.

[15] C. Spampinato, S. Palazzo, I. Kavasidis, D. Giordano, N. Souly, and M. Shah, "Deep learning human mind for automated visual classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6809–6817, doi: 10.1109/CVPR.2017.479.

[16] I. M. Pires, N. M. Garcia, N. Pombo, F. Flórez-Revuelta, S. Spinsante, and M. C. Teixeira, "Identification of activities of daily living through data fusion on motion and magnetic sensors embedded on mobile devices," *Pervas. Mobile Comput.*, vol. 47, pp. 78–93, Jul. 2018, doi: 10.1016/j.pmcj.2018.05.005.

[17] A. K. M. F. Haque, M. H. Ali, and M. A. Kiber, "Improved spectrogram analysis for ECG signal in emergency medical applications," *Int. J. Adv. Comput. Sci. Appl.*, vol. 1, no. 3, pp. 39–44, 2010. [Online]. Available: https://www.ijacsa.thesai.org

[18] M. Li *et al.*, "Scaling distributed machine learning with the parameter server," in *Proc. 11th USENIX Symp. Oper. Syst. Design Implement.*, 2014, pp. 583–598.

[19] J. Dean *et al.*, "Large scale distributed deep networks," in *Proc. Int. Conf. Neural Inf. Process. Syst. (NIPS)*, Dec. 2012, pp. 1223–1231, doi: 10.5555/2999134.2999271.

[20] E. P. Xing *et al.*, "Petuum: A new platform for distributed machine learning on big data," *IEEE Trans. Big Data*, vol. 1, no. 2, pp. 49–67, Jun. 2015, doi: 10.1109/tbdata.2015.2472014.

[21] Z. Chen *et al.*, "Deep learning research and development platform: Characterizing and scheduling with QoS guarantees on GPU clusters," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 1, pp. 34–50, Jan. 2020, doi: 10.1109/TPDS.2019.2931558.

[22] J. J. Dai *et al.*, "BigDL: A distributed deep learning framework for big data," in *Proc. ACM Symp. Cloud Comput.*, 2019, pp. 50–60, doi: 10.1145/3357223.3362707.

[23] M. A. Alsheikh, D. Niyato, S. Lin, H.-P. Tan, and Z. Han, "Mobile big data analytics using deep learning and apache spark," *IEEE Netw.*, vol. 30, no. 3, pp. 22–29, May 2016, doi: 10.1109/MNET.2016.7474340.

[24] B. Hu, E. Rouse, and L. Hargrove, "Fusion of bilateral lower-limb neuromechanical signals improves prediction of locomotor activities," *Frontiers Robot. AI*, vol. 5, p. 78, Jun. 2018, doi: 10.3389/frobt.2018.00078.

[25] 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, doi: 10.1093/gigascience/giy098.

[26] Z. Peng, C. Cao, Q. Liu, and W. Pan, "Human walking pattern recognition based on KPCA and SVM with ground reflex pressure signal," *Math. Problems Eng.*, vol. 2013, pp. 1–12, Dec. 2013, doi: 10.1155/2013/143435.

[27] K. Amanpreet, "Machine learning-based novel approach to classify the shoulder motion of upper limb amputees," *Biocybernetics Biomed. Eng.*, vol. 39, no. 3, pp. 857–867, Jul. 2019, doi: 10.1016/j.bbe.2019.07.007.

[28] S. Pancholi and R. Agarwal, "Development of low cost EMG data acquisition system for arm activities recognition," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Sep. 2016, pp. 2465–2469, doi: 10.1109/ICACCI.2016.7732427.

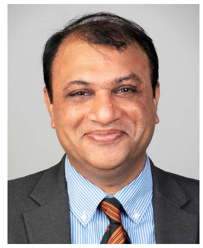
[29] S. Ranasinghe, F. Al Machot, and H. C. Mayr, "A review on applications of activity recognition systems with regard to performance and evaluation," *Int. J. Distrib. Sensor Netw.*, vol. 12, no. 8, pp. 1–22, Aug. 2016, doi: 10.1177/1550147716665520.

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. He is currently working as an Associate 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.

Tufail Khan received the M.Sc. degree from the Department of Computer Science, University of Peshawar, Pakistan, in 2007. He is currently pursuing the M.S. degree with the Department of Computer Science, Abdul Wali Khan University, Mardan, Pakistan. His research interests include machine learning techniques for activity recognition, big data analysis, parallel computing, and pattern recognition.

Mukhtaj Khan received the Ph.D. degree from the Department of Electronic and Computing Engineering, Brunel University London, U.K., in 2015. He was a Postdoctoral Research Fellow with the Department of Electronic and Computing Engineering, Brunel University London, from 2015 to 2016. He is currently working as an Assistant Professor with the Department of Computer Science, Abdul Wali Khan University, Mardan, Pakistan. His research interests include performance modeling, big data analytics, parallel computing, machine learning techniques, and underwater sensor networks.

Tahir Hussain received the M.S. degree in computer science from International Islamic University, Islamabad, Pakistan, in 2011, and the Ph.D. degree in computer science from the Department of Computer Science, Abdul Wali Khan University, Mardan, Pakistan, in 2020. His research interests include pattern recognition, biological signal processing, and human activity recognition.



Tahir Hameed received the master's degree in computer science from the Lahore University of Management Sciences (LUMS) and the Ph.D. degree in information technology management from the Korea Advanced Institute of Science and Technology (KAIST). He was a Faculty Member with the SolBridge International School of Business, South Korea, from 2012 to 2018. He has been an Assistant Professor of Management Information Systems with the Merrimack College since 2018. His current research interests include health analytics particularly clinical decision support systems and consumer health informatics, IT standards, IT adoption, and technology innovation and commercialization.

Syed Ahmad Chan Bukhari (Senior Member, IEEE) received the Ph.D. degree in computer science from the University of New Brunswick, Canada. Then, he went on to complete his Postdoctoral Fellowship at the Yale School of Medicine, where he worked with the Center of Expanded Data Annotation and Retrieval (CEDAR), Stanford University, to develop data submission pipelines to improve scientific experimental reproducibility. He is as an Assistant Professor and the Director of Healthcare Informatics at St. John's University, New York. His current research efforts are concentrated on addressing several core problems in the area of healthcare informatics and data science. He particularly focuses on devising techniques to semantically confederate heterogeneous biomedical data and to further develop clinical predictive models for diseases prediction. These techniques further alleviate many data access-related challenges faced by healthcare providers. He is a Distinguished ACM Speaker, who serves as an editorial board member for multiple scientific journals. His research work has published in top-tier journals and picked by various scientific blogs and international media.