A Data-Driven Approach to Kinematic Analytics of Spinal Motion

Ayşenur Gençdoğmuş
Department of Computer Engineering
Sabahattin Zaim University
Istanbul, TURKEY
aysenur.gencdogmus@izu.edu.tr

Şeref Recep Keskin
Department of Computer Engineering
Gazi University
Ankara, TURKEY
serefrecepkeskin@gmail.com

Gülüstan Doğan
Department of Computer Science
University of North Carolina Wilmington
North Carolina, NC, USA
dogang@uncw.edu

Yusuf Öztürk
Department of Electrical and Computer Engineering
San Diego University
San Diego, USA
yozturk@sdsu.edu

***Abstract*—A common reason for low back pain can be attributed to postural stress. While seated or walking, bad posture can put strain on the spine. Increased stress on the spine may induce tightness and spasms in the lower back muscles and may lead to low back pain. Recognition of unstructured daily activities becomes a more difficult and essential task, as monitoring of daily activities becomes more important, especially for helping sick and elderly people. In this study, we employ deep learning and machine learning methods to study spine motion and postural stress using two sensors attached to lower back of a healthy subject while the subject is performing regular daily activities. A comparison of the accuracy of deep learning and supervised machine learning approaches (Decision Tree, Random Forest, Gradient Boosting, AdaBoost, KNN, Naive Bayes) in identification and labeling of daily activities is provided. In addition, the effective values for hyper parameters of LSTM neural networks have been determined. LSTM neural networks achieved highest accuracy.**

***Index Terms*—Big Data Analytics, Data Mining, Machine Learning, Deep Learning, LSTM Neural Networks**

I. Introduction

Human activity recognition has been a widespread research topic, especially in the recent years. Recording human activities without restricting the movement of people is important. It is also not possible to give precise information about the duration of spine posture and the frequency of movement of functional activities in a single session. Taking all these limitations into consideration, the patient's movement data were evaluated with the following: to monitor the position and movement of the spine, a study at the University of San Diego [1] is performed using two wireless body sensors.

This study is a continuation of the study at the University of San Diego. It aims to coordinate data from wireless body sensors attached to the subject's pelvis and spine. The body kinematics monitoring data comes from the 9-axis motion processors of the sensors. This data is transmitted wirelessly via Bluetooth Low Energy (BLE) to an Android device at a sampling frequency of 100 Hz.

There is noise in the dataset due to problems caused by both the environment and the sensors because the body kinematics data have been collected in the natural environment of a person. Parameters used in datasets are; Sequence Number, MAC ID of the sensor, Time Stamp, Position Number, Quaternion X, Quaternion Y, Quaternion Z, Quaternion W. First, we matched the data from two different sensors and then we converted the data to the appropriate format. After these steps, we applied a median filter to eliminate the noise in the data. We used R programming language for all calculations, visualizations, and also matching and data filtering processes. After completing the data preprocessing step, we applied deep learning method and machine learning methods. We used Python programming language to implement these methods.

Our study aims to analyze the human activity data obtained from body sensors. After data preprocessing step, we used machine learning and deep learning methods. LSTM neural networks method and supervised machine learning methods (Decision Tree, Random Forest, Gradient Boosting, AdaBoost, KNN, Naive Bayes) were used to analyze and compare the results. In addition to the comparison of machine learning and deep learning methods, some hyperparameters used in the deep learning method were tested and the effective values of these hyperparameters were determined.

II. Related Work

In this section, we will review the work that is related to our study, and list our contributions.

Recently, Deep learning has shown promising performance in many areas with automatic high-level feature extraction. Summarizing the current literature from three perspectives, this study investigates the sensor method, deep models and applications. It also considers the latest developments in the field of sensor-based activity recognition based on deep learning [2].

978-1-7281-0858-2/19/$31.00 ©2019 IEEE

New ideas have come into view to address the Human Activity Recognition (HAR) problems that had emerged with the development of deep learning. One is to propose a deep network architecture that uses bidirectional residual (incremental) Long Short-Term Memory (LSTM) cells [3]. The advantage of this proposed method is that a two-way connection can combine the positive time direction (forward state) and the negative time direction (reverse state).

In another study, it is considered that recurrent neural networks (RNNs) with long Short Term Memory (LSTM) can automatically learn and model long-term transient dependencies [4]. In this study, the end-to-end fully connected LSTM network for skeletal-based action recognition is proposed. The experimental results in three human action recognition datasets consistently demonstrated the effectiveness of the proposed model.

In the study of other researchers, deep, convective and repetitive approaches are investigated in three representative datasets. The datasets contain motion data captured by wearable sensors. Subjects like how to train repetitive approaches, how to introduce a new regularization approach, and how the most advanced technology performs better in a state-of-the-art dataset are explained. For this purpose, LSTMs containing multiple repetitive unit layers are used [5]. Similar to this study, in our study we apply a multi-layer LSTM model and compare contribution of this number of layers to the learning of the model.

Unlike traditional LSTM, a method is developed where information is shared between multiple LSTMs with a new pool layer. This pool layer works by combining the hidden representation of LSTMs corresponding to neighboring trajectories [6]. The performance of the method is shown in several public datasets. This model was found to perform 42% better than previous prediction methods.

Researchers aiming to use deep learning approaches for activity recognition observed the effect of various important hyperparameters, such as convoluted layer number and core size, on the performance of CNN [7]. Similar to this study, in our study we also show the effects of hyperparameters such as batch size, LSTM layer number and epoch number used in the LSTM network on accuracy.

A system called Real-Time Activity Monitoring (RAM) is developed to monitor real-time continuous activity. RAM performs multi-class classification with the SVM method and compares it to 8 different configurations in which the SVM and Nearest Neighbor (KNN) classifiers are fed with predefined features [8]. In our study, we use the KNN classifier which is one of the methods of supervised machine learning. We demonstrate its difference with other classifiers.

In another study, combining data from two wearable inertial sensors connected to one foot and one waist (similar to the two wireless body sensors placed in the pelvis and spine in our study) is suggested to evaluate a person's daily activities. [9].

III. DATA ANALYTICS

In this section, we will present the process of cleaning, transforming, visualizing and filtering body kinematics data.

In this study, we used R programming language for all operations on the data through RStudio. The collection of data from two wireless body sensors affixed to the human body was performed through a mobile application.

The data coming from the sensors were saved in .csv format. Verification data collected in the laboratory was used before processing the ecological data. The laboratory data was used to classify ecological data.

A. DATA PROCESSING

1) Data Cleaning-Grouping: In body kinematics data, there are noises caused by android mobile application and environmental factors. Noise can be in the form of missing, inaccurate or duplicate data. The first process to eliminate the noise in the data is to remove repetitive data from the dataset. Then the data from two different sensors is grouped.

2) Sensor Data Matching: Since the aim of our study is monitoring the movement of the sensors with respect to each other, first we match the data that we divide into two different groups. Under normal conditions, we expect the data to come in sequence from both sensors, but we have seen that the data does not always come in order, either due to environmental reasons (noise, disturbance of the frequency, everything that enters the sensor and loss of connectivity) or problems caused by the sensor. Therefore, we use sequence numbers (Seq_Num) of the incoming data in order to match the data that is not consecutively in sequence. Algorithm 1 is the pseudo-code that matches the sensor readings.

Algorithm 1 Sensor Data Matching

```
for i=1:length (data) do
  a ← data[i]
  if a ∉ data[i] then
    for j = 1:5 do
      b ← a − j
      if a ∈ data[i] then
        break
      else
        b ← a + j
        if b ∈ data[i] then
          break
        end
      end
    end
  end
end =0
```

Quaternions [10] are mathematical tools that can be converted to various other geometric expressions such as rotational matrices, rotational axes and angles, spherical rotational angles and Euler rotational angles. While the sensor pairing is

performed in Algorithm 1, the quaternion values are converted to the rotation matrix for each matching record.

3) Calculation of Rotation Matrix: After matching the sensor data given in pseudo-code in Algorithm 1, we have to perform several matrix operations on the matrix to convert the rotation matrix to Euler angles. The first step is taking the symmetry of the rotation matrix to create a mirror image. Then, the order of the rows is changed. The row order of the matrix with the order of 1-2-3 is rearranged to 3-2-1. Since some elements of the matrix must be negative, the elements to be replaced are multiplied by -1 as the third process. The rotation matrices are 3x3 dimensional matrices. Since the function to be used in calculating Euler angles requires operating with 4x4 dimensional matrices, new rows and columns are added to the rotation matrix.

The pseudo-code that shows the calculation of the rotation matrix and the operations on the matrix is as in Algorithm. 2.

Algorithm 2 Calculation of The Rotation Matrix and The Operations on The Matrix

```
0: rotMatrix ← quaternion2rotation(quat values)
0: mirror (rotMatrix)
0: change row order of rotMatrix as (3, 2, 1)
0: add a row to rotMatrix as (0,0,0)
0: add a column to rotMatrix as (0,0,0,1) =0
```

4) Conversion of Rotation Matrices to Euler Angles: After all the operations on the matrix are completed, Euler angles are calculated. We perform the conversion in the R programming language using the "rot2eul" function in the directional package. Since the function calculates angles in radians, a function that converts radians to degrees is used.

In Algorithm 3, the pseudo-code that calculates Euler angles and converts them into degrees is given.

Algorithm 3 Calculation of Euler Angles

```
0: eulangles ← rot2eul (rotMatrix)
0: as matrix eulangles
0: rad2deg (eulangles)
0: determine eulangles dimension as 1x3 =0
```

In our study, since the motion of the two sensors is examined, it is necessary to calculate the values of the matching data after matching the sensor data. This calculation is made by taking the difference of angles.

B. DATA VISUALIZATION

1) Visualization of Noisy Data: Before any filtering process has been performed on the raw data from the sensors, we plotted the noisy data. We used the "ggplot2" package for advanced graphs.

"ggplot2" package: ggplot2 is a data visualization package for the statistical programming language R. While visualizing data, this package separates graphs into semantic components such as scales and layers. Examples of the visualized positions are given in Figure1.

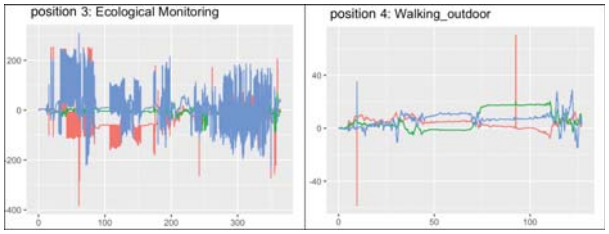

Fig. 1. Graph of recorded (noisy) positions during ecological monitoring and walking outdoor activities

2) Filtering Noisy Data: We first removed the "NA" values in the dataset because there was noisy or missing data in the values recorded from the sensors. There are also incorrect values in consecutive records in the dataset. We applied "Median Filter" in the dataset to solve the problem.

Median Filter: The Median Filter is a non-linear digital filtering technique, often used to remove noise from an image or signal. Such noise reduction is a typical pre-processing step to improve the results of later processing (for example, edge detection on an image) [11]. Median filtering is very widely used in digital image processing because, under certain conditions, it preserves edges while removing noise (but see discussion below), also having applications in signal processing.

The main idea of the median filter is to input by signal input, replacing each input with the median of neighboring inputs. The pattern at neighboring values is called a "window". The number of windows is calculated by the equation 1:

$$k < -1 + 2 \times min\left((nrow\,(\text{data}) - 1)\,\%\,/\,\%2, ceiling\right.\\ \left.(0.1 \times nrow\,(\text{data}))\right) \quad (1)$$

3) Visualization of Filtered Data: The number of windows to be used in the median filter was calculated and applied as 81 according to the form given in equation 1. After applying the filter, extremum points in the dataset were subtracted and the noise was removed.

The graphs of the filtered positions measured during ecological monitoring and walking outdoor activities for Euler angles on the x, y, z axes are as shown in Figure 2 below. We showed unfiltered and filtered graphical visuals and compared the median filter result. The graphs on the left of the figure are obtained from raw data without median filter applied and the graphs on the right are graphs drawn with median filtered data.

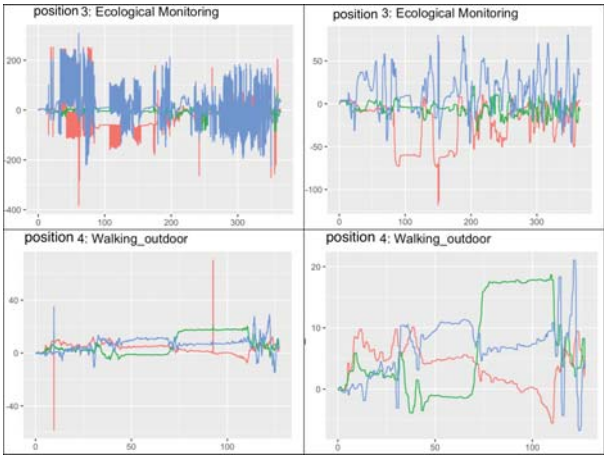

Fig. 2. Filtered and unfiltered graphs for ecological monitoring and walking outdoor positions

IV. MACHINE LEARNING METHODOLOGY

In this section, we briefly describe the machine learning algorithms that had been used for our experiments.

In this study, everything was done in the Python programming language. Jupyter Notebook and Spyder technologies were used in the anaconda development environment. In the machine learning process scikit-learn, matplotlib, numpy and pandas libraries were used.

Experiments have been made on machine learning algorithms using this dataset. Decision Tree, Random Forest, Gradient Boosting, AdaBoost, KNN, Naive Bayes algorithms were used in the experiments. 70% of the 95978 samples in this dataset were used to test the remaining 30% of the model to train the model. We compared the machine learning results using the six different machine learning models. The model's funcitons are taken from the scikit-learn library.

A. Naive Bayes Algorithm:

The Naive Bayes algorithm is a probabilistic mathematical approach found by Thomas Bayes. It aims to determine the category of data that is computed according to the probability principles of an event. For Naive Bayes, each parameter must be statistically independent of each other. Calculates the probability calculation according to the given data and classifies them. The class is estimated by making operations according to the probability calculations created over the incoming data. The parameter estimation for Naive Bayes models uses the maximum probability method.

The general formula of the Naive Bayes model is given below:

$$P(A/B) = \frac{P(A) * P(B/A)}{P(B)}$$

- P(A/B); Possibility of occurrence of event A when event B occurs
- P(B/A); Possibility of occurrence of event B if event A occurs
- P(A) and P(B); A and B events are the primary possibilities

The primary possibility of A and B adds subjectivity to the Bayesian theorem. It is possible to have information about probability A without learning about the event A. Once the data has started to be collected, it is possible to estimate the likelihood of B occurring in cases where A has occurred. Besides, one of the biggest advantages of Naive Bayes is that the small dataset is sufficient to make the classification.

When we apply the Naive Bayes algorithm in our dataset, it gives a success rate of 45.8%.

B. KNN Algorithm:

KNN is a method used for classification and regression. KNN aims to find the nearest neighbor. The KNN classification locates the nearest neighbors by calculating the distance of the given data to the previously taught data. The class in which the neighbors are the majority determines the outcome of the sent data. The KNN regression is the process of determining the class of data sent by taking the average of the nearest neighbor values.

KNN uses 3 different distance formulas when calculating the distance. These are Euclidean, Manhattan and Minkowski theorem. Distance calculation formulas are given below.

- Euclidean distance formula is:

$$\sqrt{\sum_{i=1}^{N}(x_i - \bar{y_i})^2}$$

- Manhattan distance formula is:

$$\sum_{i=1}^{N}|x_i - \bar{y_i}|$$

- Minkowski distance formula is:

$$\sum_{i=1}^{N}((|x_i - \bar{y_i}|)^q)^{1/q}$$

It is a computationally heavy method because it takes a long time to calculate the distances to each neighbor. It gives the closest and correct result among the classification types. It should be preferred for small datasets because of the long duration for large data. It is the most resistant algorithm to be noisy.

In our dataset, the KNN algorithm provides a success rate of 49.5%.

C. Random Forest Algorithm:

Random Forest is the machine learning method used in classification and regression. Random Forest aims to produce appropriate models for the given data by creating more than one decision tree. It obtains the average estimates of the trees generated. It selects the most suitable tree based on the average estimates.

The understanding and interpretation of Random Forest is quite simple. The process of pre-processing the tree is performed quickly. Making a simple tree makes progress through it. It is possible to achieve successful results using very few datasets. It can be used to process both numeric and class data types. Some machine learning types cannot be used for class data, while random forest trees can be used for these species. Calculation procedures are short. It produces fast results. Even if there is a large amount of data, the time to process the data and produce results is very short.

All the trees can go in different directions with an error during learning. Because of this reason, the dataset should be selected more carefully. Otherwise, wrong and different results can be produced due to unsuccessful training, different models may be encountered.

Random Forest algorithm gives a success rate of 48.5% in our dataset.

D. Adaboost & Gradient Boosting Algorithm

The most general definition of boosting algorithms is to turn a weak learning model into a powerful model. It aims to combine several weak learner models into a strong model. In other words, it aims to make the success of the model better by training each other in succession. The most popular algorithms are AdaBoost and Gradient Boost.

In our study, Gradient Boosting achieved 47% success while Adaboost achieved 13.7% success.

E. Decision Tree Algorithm

Decision tree algorithm is one of the tree-based learning algorithms. Also it is one of the methods of supervised machine learning. In this method, datasets containing a large number of records are divided into smaller clusters by performing simple decision-making steps. This clustering continues in depth until all elements of the group have the same class label. Decision tree algorithms are basically divided into two categories: Entropy classification trees (ID3, C4.5) and Regression trees (CART).

In our study, the Decision Tree algorithm gives a success rate of 45.7% in our dataset.

V. DEEP LEARNING METHODOLOGY

After cleaning, filtering and visualization methods were applied on body kinematics data, deep learning method which is a machine learning method was applied.

Deep learning uses artificial neural networks. Artificial neural networks, like the human brain, consists of neurons. All neurons are interconnected and affect output. Neurons are basically composed of 3 layers:

1. Input Layer: This is the layer where the input data is located. It passes the input data to the first hidden layer.

2. Hidden Layer (s): This layer performs mathematical calculations on our inputs. One of the difficulties in creating artificial neural networks is to decide the number of hidden layers as well as the number of neurons for each layer. In deep learning, "deep" refers to having more than one hidden layer.

3. Output Layer: This is the layer on which output data is created.

Deep learning deals with the prediction of output data in the output layer. One of the most difficult parts of learning is to train the neural network because deep learning requires a huge amount of data and a large amount of computational power.

In this study, an LSTM (Long Short-Term Memory) Neural Network training applied in TensorFlow was applied for body kinematics monitoring data obtained from the sensors data. The general scheme of the LSTM neural network is shown in Figure 3. In this LSTM scheme, blue-colored boxes perform scoring and yellow-colored shapes represent layers.

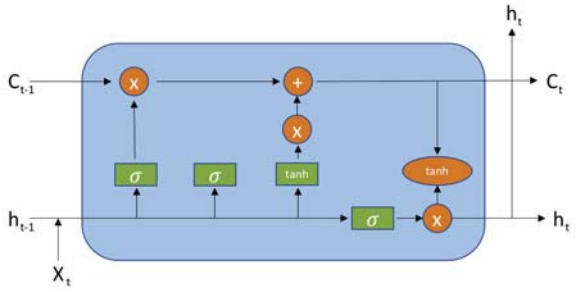

Fig. 3. LSTM neural network scheme [12]

A. Application of Deep Learning on Body Kinematics Data

In this study, the Python programming language was used for deep learning applications. The main libraries are numpy, pandas, TensorFlow and seabor

First of all, the numbers of the positions were added to the "class" column we created numerically in the dataset that we used in our study. In addition, the names of the position numbers have been added to the newly created activities column. The edited dataset is as in Figure 4.

Fbend_rep	7	0.061	-0.0026774571	0.006780465	-0.04978697
Fbend_rep	7	0.090	-0.0004084963	0.103928704	-0.09517221
Fbend_rep	7	0.091	0.0005027139	0.106410099	-0.09758568
Fbend_rep	7	0.119	0.0036351639	0.107317985	-0.09829972
Fbend_rep	7	0.120	0.0036351639	0.107502550	-0.10195900
Fbend_rep	7	0.209	0.0036351639	0.107317985	-0.10726079
Fbend_rep	7	0.211	0.0005027139	0.106410099	-0.10726079
Fbend_rep	7	0.270	0.0003271739	0.107317985	-0.10780485
Fbend_rep	7	0.271	0.0005027139	0.107502550	-0.10924501
Fbend_rep	7	0.272	0.0033384701	0.107850691	-0.10949979
Fbend_rep	7	0.299	0.0036351639	0.107883800	-0.11082485
Fbend_rep	7	0.300	0.0036351639	0.107896756	-0.11222430

Fig. 4. Reorganized (filtered) body kinematics data for the Deep Learning model

In the model used, the number of training rounds (epoch), which is defined as the time when an entire dataset is passed

both back and forth in the neural network, was initially tried as 50 and then the results were compared by increasing the number of training rounds. The definition of the parameters and hyperparameters of the Deep Learning method is as in Figure 5.

```
# Data preprocessing
TIME_STEP = 100

# Model
N_CLASSES = 29
N_FEATURES = 3  # x-acceleration, y-acceleration, z-acceleration

# Hyperparameters
N_LSTM_LAYERS = 2
N_EPOCHS = 100
L2_LOSS = 0.0015
LEARNING_RATE = 0.0025

# Hyperparameters optimized
SEGMENT_TIME_SIZE = 180
N_HIDDEN_NEURONS = 30
BATCH_SIZE = 32
```

Fig. 5. Defining the parameters and hyper parameters of Deep Learning method

B. Generating of LSTM Neural Network

When creating the LSTM neural network, the input layer contains the stack size, each segment size of the segmented data, and the number of properties in the dataset.

While creating the weights of hidden layers in the LSTM neural network, random values were generated by applying a normal distribution between the number of features and the number of hidden neurons in the dataset.

While creating the output layer weights in the LSTM neural network; Random values were generated by applying normal distribution between the number of hidden neurons in the dataset and the number of classes in the dataset. When constructing the hidden bias of the LSTM neural network, random values were generated such that the mean number of hidden neurons is 1 (mean = 1). Similarly, while generating the output bias value of the LSTM neural network, random values were generated by applying the normal distribution for the number of classes in the dataset.

In this model, where the number of LSTM layers is determined as 2, two LSTM cells are joined together to form layers. Finally, single-output was obtained from multiple inputs and its output in the neural network was reached. Our model includes 2 fully connected and 2 LSTM layers (stacked on each other), each containing 64 units.

C. Creating Deep Learning Model

The LSTM model we created in our study expects fixed-length arrays as training data. A known method was used to produce them. Each series contains 100 training examples. 30% of the dataset is test data; 70% were used as training data.

In our model, input and output tensors are defined separately. We should name the tensor appropriately for output estimates. The L2 regulator will be used and the regulator that we use should be specified in the loss. When creating the model, the accuracy parameters are defined and AdamOptimizer is defined as an optimizer.

AdamOptimizer: AdamOptimizer is an optimization algorithm that can be used instead of the classical stochastic gradient landing procedure to update recursive network weights on training data. AdamOptimizer differs from classical stochastic gradient descent. This method calculates individual learning rates for different parameters from the estimates that the gradients subtract from the first and second values.

Some hyperparameters used in the method are given below:

- **Epoch:** Since the training model is not trained at once, weights are updated to be retrained each time. Weight is calculated from the beginning of each new training data being trained. In this way, the most suitable weight values for the model is calculated. Each of these steps is called epoch.
- **Batch Size:** It can be defined as the memory that the model uses when applying the model. 2 and its multiples (4,8,16,32,64, ..., 512).
- **Number of Layers:** The most important feature that distinguishes the deep learning method from other artificial neural networks, especially in complex problems, is the number of layers.

VI. EXPERIMENTS

In this section, we describe the results of the experiments that are conducted in our study.

A. Deep Learning Method Tests

1) Effect of LSTM Layer Number on Accuracy: In this section, it is aimed to see the effect of the change in the LSTM layer number on the accuracy of the test. One of the most important features that differentiate the deep learnin method from other artificial neural networks, especially in complex problems, is the number of layers. In this direction, the tests were conducted by increasing the number of LSTM layers with the number of training rounds being 100 and the batch size (batch_size) being 32. All positions were used in the tests and the number of classes (N_CLASSES) in the model was determined as 29.

Five tests were performed to see the effect of the LSTM layer number on accuracy and loss. In the first test, the LSTM layer number was determined as 2 and in the other tests, tests for 4,6,8 and 10 layers were performed.

The train and test data in the LSTM neural network were formed by the model as 70% training data and 30% test data.

When we look at the graph in Figure 6, increasing the number of LSTM layers for the body kinematics dataset used in our study decreased the success of the model and accuracy.

2) Effect of Epoch Number on Accuracy: In this section, it is aimed to see the effect of the change in epoch number on the accuracy of the test. In these tests, the batch size (batch_size) will be kept constant at 32 and the LSTM layer number will be determined as 2 since the best results are obtained in the 2-layer LSTM model. Again all positions will be used, the

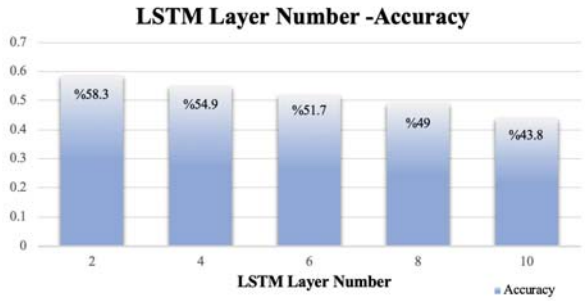

Fig. 6. Effect of LSTM layer number on accuracy graph

number of classes in the model (N_CLASSES) will be 29. In this way, it is planned to see the effect of the number of training rounds (epoch) on accuracy and loss.

Five tests were performed to see the effect of layer number on accuracy. In the first test, the epoch number was determined as 50 and in the other tests, tests for 100,150,200 and 250 epoch number were performed.

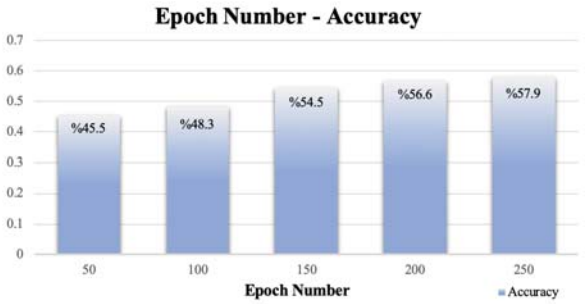

Fig. 7. Effect of epoch number on accuracy graph

When we look at the graph in Figure 7, increasing the epoch number for the body kinematics dataset used in our study increased the success of the model and increased accuracy. The high number of epochs in deep learning methods has a positive effect on the success of the model in general, but increasing it continuously does not always increase the success of the model. If the epoch number increases continuously, accuracy may decrease due to overfitting.

3) Effect of Batch Size on Accuracy: In this section, it is aimed to see the effect of the change in batch size on the accuracy of the test. According to the results obtained from previous tests, it was observed that the highest accuracy value was obtained when the epoch number was 250. For this reason, the epoch number of the tests in this section will be kept constant at 250, while the LSTM layer number, since the best results are obtained in the 2-layer LSTM model, will be determined as 2 and the batch size will be increased in other tests. Again all positions will be used, the number of classes in the model (N_CLASSES) will be 29.

Because the batch size must fit in the GPU memory, the batch size value must be set to 2 and multiples (2, 4, 8, 16,..., 512). If not determined in this way, there may be sudden decreases in the success of the model. It is usually between 64 and 512. In our study, there are five tests and batch size values starts from 32 up to 512 in multiples of 2. (32, 64, 128, 256 and 512)

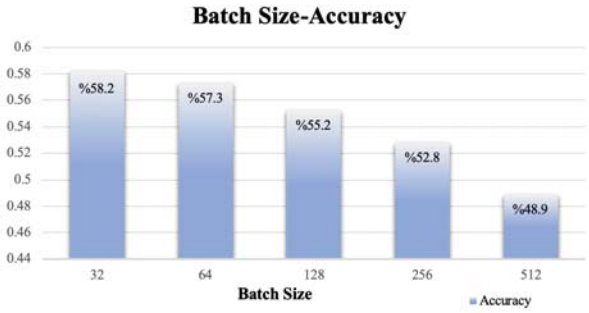

Fig. 8. Effect of batch size on accuracy graph

When we look at the graph in Figure 8, it is seen that educating the LSTM neural network with a smaller batch size provides higher success. The very high definition of batch size negatively affects the success of the model.

B. Machine Learning Methods Tests

In this study, the results of the models were tested by comparing body kinematics data on DecisionTree, RandomForest, GradientBoosting, AdaBoost, KNN, Naive Bayes methods. The training and test datasets to be used in the methods in this section are determined as 70%-30%.

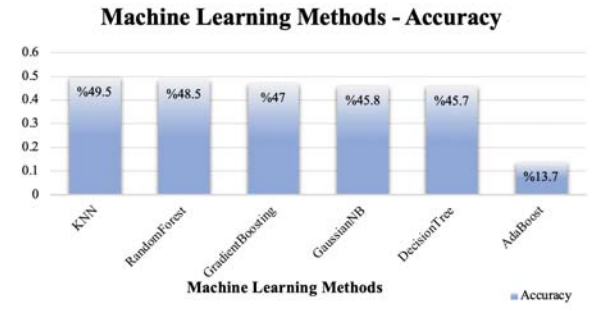

Fig. 9. Machine Learning Methods Accuracy

As shown in Figure 9, although the differences in the accuracy of machine learning methods are very large, the highest accuracy rate was obtained with the KNN method. However, the AdaBoost method has been found to have a very low accuracy percentage, and it was not found suitable for body kinematics monitoring data.

VII. CONCLUSION

In this study, we compared the results of deep learning model and machine learning methods in recognition of human activity. Data analysis were performed using the data collected via two wireless body sensors attached to a person's body. After completing the processes of cleaning data, calculating Euler angles, matching, and filtering, we used a median filter to eliminate the noise in the dataset. Then, we applied a deep learning method and classical supervised machine learning methods on the filtered body kinematics data.

In conclusion, after comparing the results of supervised machine learning methods (Decision Tree, Random Forest, Naive Bayes, AdaBoost, GradientBoosting, KNN) and LSTM neural network method, we obtained the highest success with the deep learning method.

We found that the most effective LSTM parameter values are as follows: the layer number is 2, the batch size is 32 and the epoch number is 250.

We anticipate that deep learning methods, that had high success in the classification of human activities, will be used frequently in classification of body kinematics data in the future. For future work, we think that in addition to activity classification, sudden and unexpected changes and abnormalities in activities can be analyzed. Using the results of these analyses along with their interpretation by experts, we plan to facilitate the treatment of sick and elderly people.

ACKNOWLEDGEMENT

We would like to thank Merve Kayhan, Lara Yener and Alyssa Yesilyurt for proofreading the article. This study is supported by the Yildiz Technical University Scientific Research Project (BAP) numbered 3539 named Motion Kinematics Data Analysis.

REFERENCES

[1] P. Paladugu, A. Hernandez, K. Gross, Y. Su, A. Neseli, S. Gombatto, K. Moon, and Y. Ozturk, "A sensor cluster to monitor body kinematics," in *2016 IEEE 13th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*. IEEE, 2016, pp. 212–217.

[2] J. Wang, Y. Chen, S. Hao, X. Peng, and L. Hu, "Deep learning for sensor-based activity recognition: A survey," *Pattern Recognition Letters*, vol. 119, pp. 3–11, 2019.

[3] Y. Zhao, R. Yang, G. Chevalier, X. Xu, and Z. Zhang, "Deep residual bidir-lstm for human activity recognition using wearable sensors," *Mathematical Problems in Engineering*, vol. 2018, 2018.

[4] W. Zhu, C. Lan, J. Xing, W. Zeng, Y. Li, L. Shen, and X. Xie, "Co-occurrence feature learning for skeleton based action recognition using regularized deep lstm networks," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[5] N. Y. Hammerla, S. Halloran, and T. Plötz, "Deep, convolutional, and recurrent models for human activity recognition using wearables," *arXiv preprint arXiv:1604.08880*, 2016.

[6] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 961–971.

[7] T. Zebin, P. J. Scully, and K. B. Ozanyan, "Human activity recognition with inertial sensors using a deep learning approach," in *2016 IEEE SENSORS*. IEEE, 2016, pp. 1–3.

[8] G. Uslu and S. Baydere, "Ram: Real time activity monitoring with feature extractive training," *Expert Systems with Applications*, vol. 42, no. 21, pp. 8052–8063, 2015.

[9] C. Zhu and W. Sheng, "Human daily activity recognition in robot-assisted living using multi-sensor fusion," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 2154–2159.

[10] J. B. Kuipers *et al.*, *Quaternions and rotation sequences*. Princeton university press Princeton, 1999, vol. 66.

[11] Wikipedia contributors, "Median filter — Wikipedia, the free encyclopedia," 2019, [Online; accessed 17-August-2019]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Median$_filteroldid = 911113661$

[12] "Time series prediction using lstm deep neural networks." [Online]. Available: https://www.altumintelligence.com/articles/a/Time-Series-Prediction-Using-LSTM-Deep-Neural-Networks