

Confusion Matrix				
Output Class	running	sitting	walking	
	<div>60 19.7%</div>	<div>0 0.0%</div>	<div>2 0.7%</div>	<div>96.8% 3.2%</div>
	<div>0 0.0%</div>	<div>121 39.8%</div>	<div>0 0.0%</div>	<div>100% 0.0%</div>
	<div>0 0.0%</div>	<div>4 1.3%</div>	<div>117 38.5%</div>	<div>96.7% 3.3%</div>
	<div>100% 0.0%</div>	<div>96.8% 3.2%</div>	<div>98.3% 1.7%</div>	<div>98.0% 2.0%</div>
	running	sitting	walking	
Target Class				

Final Project

MACHINE LEARNING

Muhammad Saad, Zeynep Güvenç, Hasan Tuna Erdem | MECH 100 | 8/6/2022

Table of Contents

1. INTRODUCTION:	2
2. STEPS OF THE PROJECT:	2
2.1. Acquiring the data:	2
2.2. Transferring the data:	2
2.3. Preparing the data:	3
2.4. Model Training Process:	4
3. GRAPHED VISUALIZATIONS	4
3.1. Training Data Sequences	4
3.2. Model Training Process	5
3.3. Test data sequencing	6
3.4. Classification	7
4. ENCOUNTERED ERRORS AND FINAL NOTES	8
4.1 Speed calculation and percentage error	9
5. CONCLUSION	10

Final Project Report

1. INTRODUCTION:

In this final project we tried to execute a machine learning model to teach a machine how to guess peoples' movements and their activities based on their acceleration values. Main activities were sitting, walking and running since they were more separable from each other. For this project we used MATLAB as the base of coding and used other MathWorks products like MATLAB Drive and MATLAB Mobile to help us. We tried to get the results of our learning process from some plots that give detailed information about the distribution of acceleration values, the predicted activities and their test data.

2. STEPS OF THE PROJECT:

2.1. ACQUIRING THE DATA:

In order to get the acceleration values, we used MATLAB Mobile and the sensors in our mobile phone. We took the data at the rate of 10Hz to improve the results. We tried to make the measurements as similar as their counter data so we make 1-minute measurements for sitting, walking and running.

2.2. TRANSFERRING THE DATA:

Once we acquired the data, we needed to transfer it to MATLAB. For this step we used MATLAB Drive to transfer the data via cloud. First, we uploaded our data to MATLAB Drives' cloud then we downloaded it from the computer (This has been detailed with relevant screenshots in step 1 of the final project submission).

2.3. PREPARING THE DATA:

The raw data needed to be processed to make it usable for our learning algorithm. First, we deleted all the time stamps at the data and converted it to an array from a time table. Then we transposed the data to make it easier to visualize and to label the data (The code below was used to transpose the arrays).

```
%take the transpose of Acceleration
AccelerationSIT1=table2array(AccelerationSIT1)
AccelerationSIT1=transpose(AccelerationSIT1);
```

First, we were doing this process by hand but to make it more efficient we wrote some code to do it. After that to prepare our Test and Train matrices we divided the data to two arrays, 20 % of the data in Test and the remaining 80% in Train (Below code was utilized in doing so).

```
P = AccelerationSITTING1;
N = size(P,1);
idx = randperm(N);
AccelerationSIT1= P(idx(1:round(N*0.8)),:);
AccelerationSIT3 = P(idx(round(N*0.8)+1:end),:);
```

The split for the two Test and Train array could have been sequential with the first 80% as Train and the last 20% as Test, but the 'randperm' function was used to randomize value association in each array for a more accurate result while training the model. We made it that it doesn't choose the data at the same index twice for more precise results.

2.4. MODEL TRAINING PROCESS:

We are using sequence-to-sequence classification method for this project. No new method has been developed for model training, but an existing method has been utilized. To train our algorithm with the sequence data, we can use LSTM (long short-term memory) network, this allows us to analyze the 'time steps' (intervals of 0.1 seconds) in the sequence and make predictions for each individual time step. First, we combine the data of sitting, walking and running in order to create a larger dataset to observe. Then we label the sequences of sitting, walking and running. After that we combine the data in cell array format to make it useable for the sequence-to-sequence classification method. Then we visualize the aggregate data with plot functions on MATLAB.

3. GRAPHED VISUALIZATIONS

3.1. TRAINING DATA SEQUENCES

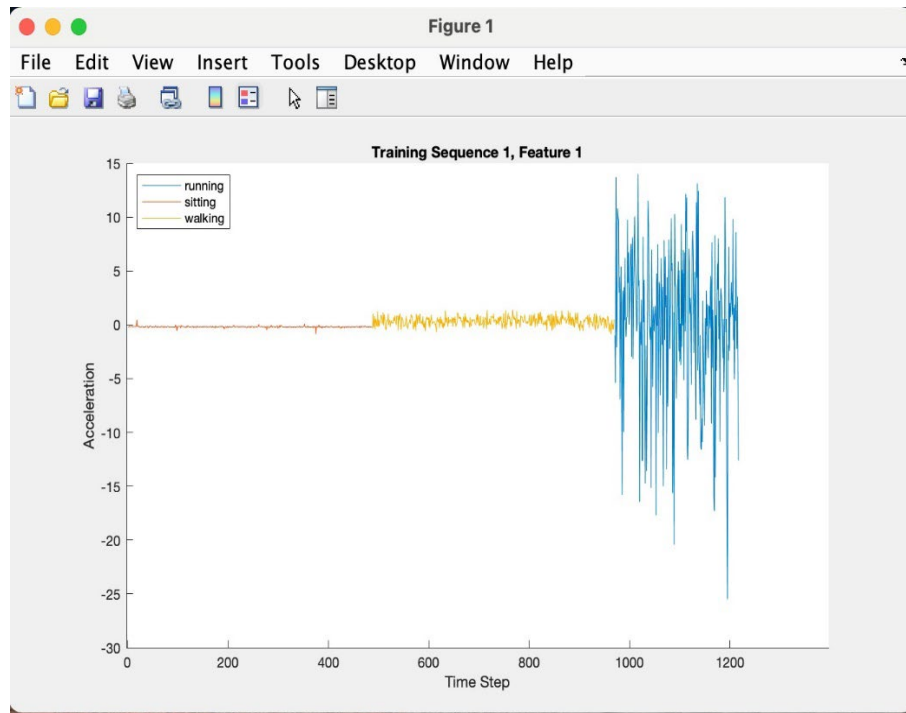


Figure 1: A graph detailing acceleration sequences for each movement type

Then we define a LSTM network architecture with 3 features and 200 hidden units. Lastly, we specify 3 classes as sitting, walking and running. After that we train the network we created with ‘trainNetwork’ function. This might take some time because the sequences are very long. MATLAB show the process with a plot.

3.2.MODEL TRAINING PROCESS

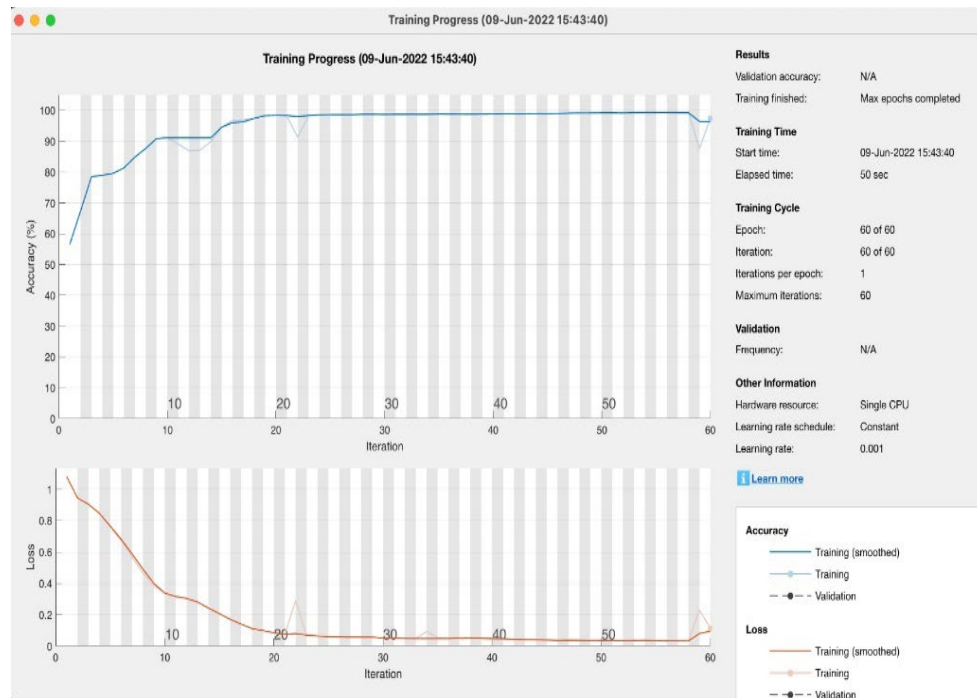


Figure 2: A graph detailing the training process for our model

Now that we have a trained algorithm, we can test this algorithm and see how accurate it is. To test the algorithm, we load the human activity test data. We can visualize the features for different sequences to better understand the differences between them. The general shape of the graph shows that loss and accuracy in the model are inter-related: as the loss decreases, the accuracy increases. Throughout the graph, the loss in the model decreases with iteration, hence leading to an increase in accuracy. However, at the end of the graph, the loss increases a bit, decreasing the accuracy of the model.

3.3. TEST DATA SEQUENCING

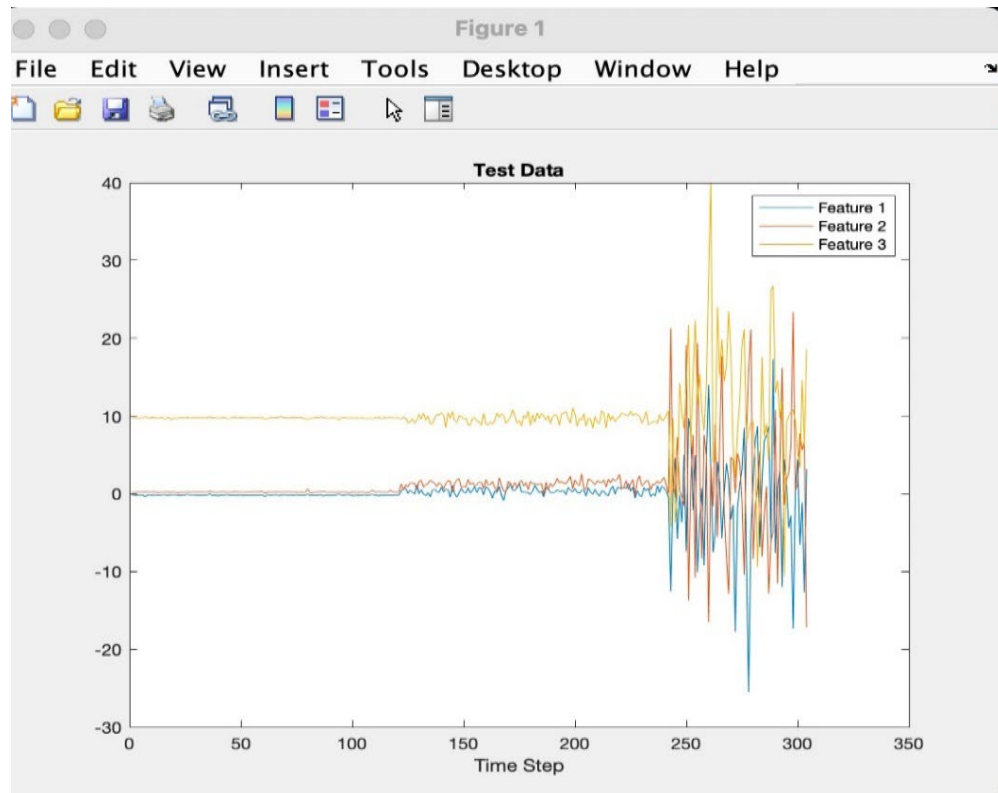


Figure 3: A graph visualizing the x, y, z acceleration sequences for each movement type

After that we classify our test data using classify function. To see the accuracy of the predictions, we can look at ratio of the correct guesses and the length of the whole sequence. This gives us a numerical value for the accuracy. Additionally, we can plot the prediction data and the test data to see our accuracy visually. The features in the graph represent the acceleration values in the x, y, and z direction. The z acceleration values are distinguishable from the x, and y values as they are the gravitational acceleration values and significantly higher than x and y values.

3.4. CLASSIFICATION

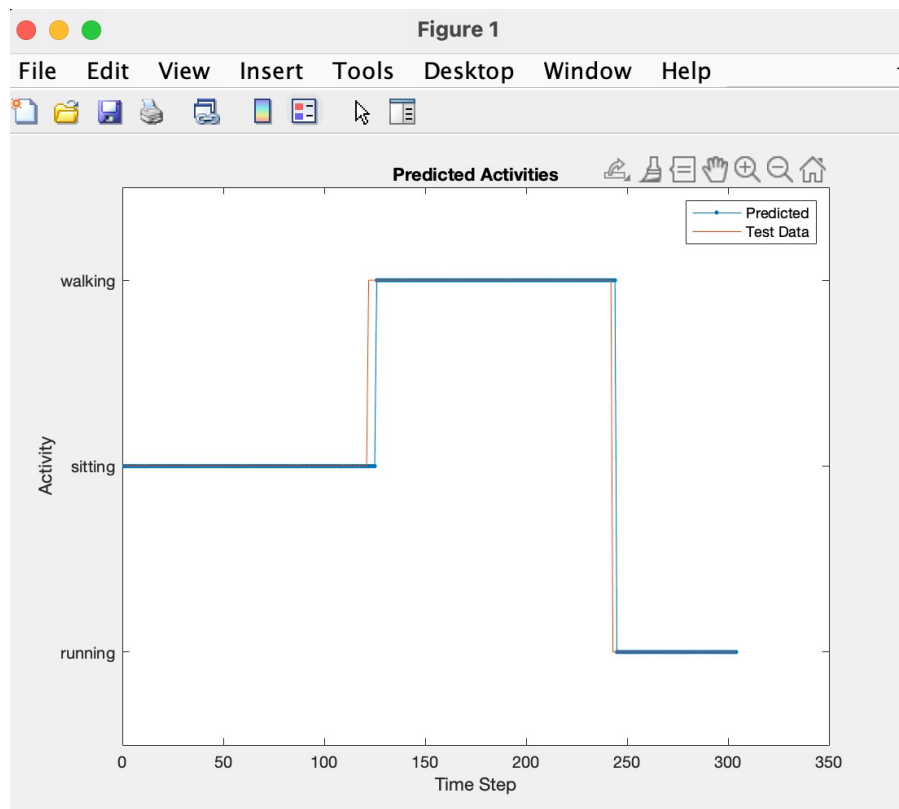


Figure 4: A graph that displays the results of our machine learning model

According to the graph, it can be seen that the model is capable of identifying test data in the proper categories. There is a very minute difference between test data, and predicted values and their classification, which demonstrates that our model is successful in performing the classification.

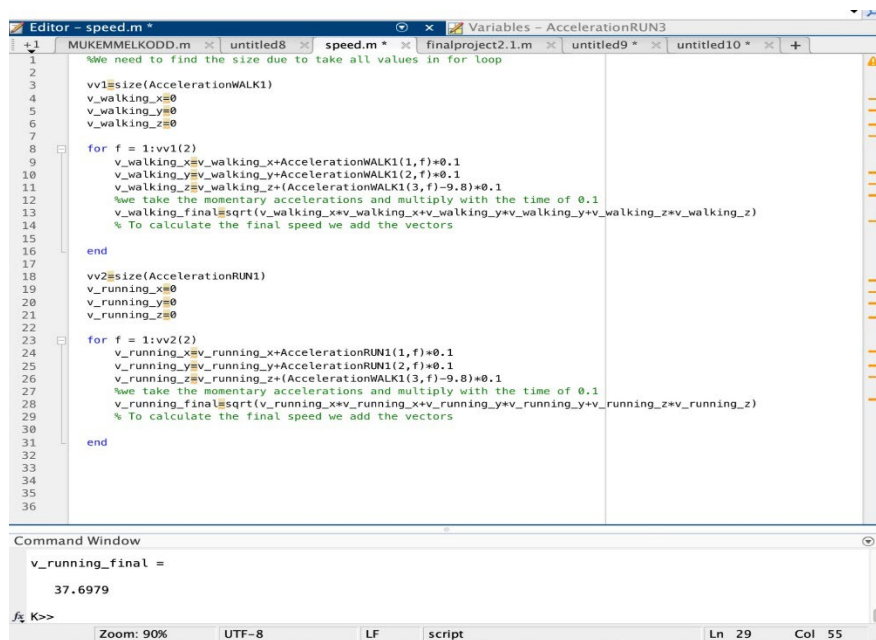
4. ENCOUNTERED ERRORS AND FINAL NOTES

We encountered several issues in recording the data. The walk and run data didn't account for centripetal acceleration when making turns which makes it unreliable to calculate velocity with a given equation. The rate at which the data was recorded (10 Hz)

was too low and resulted in less accuracy while manipulating the recorded data. The sensors built into the phone are also unreliable when gathering acceleration data. Our suggestion of dealing with these errors are to control the environment of recording data, accounting for sloping landscapes. We can also make use of a higher frequency (for instance, 100 Hz) to record the acceleration data. Dedicated sensors can be used to record the data more accurately.

4.1 SPEED CALCULATION AND PERCENTAGE ERROR

We used the following code to calculate speed. Due to errors mentioned previously, and the nature of our code (acceleration is summed up and then speed is calculated: this way the small percentage errors also add up to create a larger error), we experienced a significant percentage error for our speed values. However, the code demonstrates that the calculation is possible (attached below).



```

Editor - speed.m *
+1 MUKEMMELKODD.m x1 speed.m * x2 finalproject2.1.m x3 untitled9 * x4 untitled10 * +
1 %We need to find the size due to take all values in for loop
2
3 vv1=size(AccelerationWALK1)
4 v_walking_x=0
5 v_walking_y=0
6 v_walking_z=0
7
8 for f = 1:vv1(2)
9     v_walking_x=v_walking_x+AccelerationWALK1(1,f)*0.1
10    v_walking_y=v_walking_y+AccelerationWALK1(2,f)*0.1
11    v_walking_z=v_walking_z+(AccelerationWALK1(3,f)-9.8)*0.1
12    %we take the momentary accelerations and multiply with the time of 0.1
13    v_walking_final=sqrt(v_walking_x^2+v_walking_y^2+v_walking_z^2)
14    % To calculate the final speed we add the vectors
15
16 end
17
18 vv2=size(AccelerationRUN1)
19 v_running_x=0
20 v_running_y=0
21 v_running_z=0
22
23 for f = 1:vv2(2)
24    v_running_x=v_running_x+AccelerationRUN1(1,f)*0.1
25    v_running_y=v_running_y+AccelerationRUN1(2,f)*0.1
26    v_running_z=v_running_z+(AccelerationWALK1(3,f)-9.8)*0.1
27    %we take the momentary accelerations and multiply with the time of 0.1
28    v_running_final=sqrt(v_running_x^2+v_running_y^2+v_running_z^2)
29    % To calculate the final speed we add the vectors
30
31 end
32
33
34
35
36
Command Window
v_running_final =
37.6979
f3 K>>
Zoom: 90% UTF-8 LF script Ln 29 Col 55

```

5. CONCLUSION

Our project is a machine learning model focused on classifying acceleration values input in to the system into three different categories namely, sitting walking, and running. We made use MATLAB Mobile and MATLAB Drive to collect and import acceleration data. The data is then split with values chosen randomly from the imported timetable into time-stamped arrays at 80% for Train and 20% for Test data. We manipulated this data with the use of an existing MATLAB classification method, ‘sequence-to-sequence’ classification to train our machine learning model. Graphs were generated to show Train and Test data and the classification process as the model iterated through the data multiple times to gauge which acceleration values could be associated with relevant movement types. The model displays admirable levels of accuracy in learning train data and graphing classification results using test data which can be seen in the graphs in the report. We encountered some errors during the course of our project which we have added final notes to our report. The below graph sums up the model’s accuracy of choosing the right category for each value of acceleration data.



Figure 5: Confusion Matrix demonstrating accuracy of classification results

References

- Sequence-to-Sequence Classification Using Deep Learning - MATLAB & Simulink (mathworks.com)
- Cram.m (MECH 100 Blackboard, Course Content, Week 7)
- MatLab Smartphone Connection Tutorial (MECH 100 Blackboard, Course Content, Week of May 9)
- untitled.m (MECH 100 Blackboard, Course Content, Week of May 9)