Project 4: Wearable Gesture Recognition

| Kushal Bhatt  MCS  1211206721  Arizona State University  kvbhatt1@asu.edu | Rachel Dedinsky  MCEN  1208572075  Arizona State University  rdedinsk @asu.edu | Ashni Modi  MCS  1213185257  Arizona State University  asmodi@asu.edu |
| --- | --- | --- |

# ABSTRACT

Breakthrough in machine learning techniques and wearable sensor generalization have opened up a new era of pervasive human gesture recognition, which propels the development of relevant applications such as surveillance, virtual reality and healthcare. This project focuses on implementing MYO wristband sensor based gesture recognition application. This project is part of Spring 2018 CSE 535-Mobile Computing class. The original project methodology, algorithm and sample matlab code was shared with us by iMPACT lab. The aim is to identify user eating actions by using machine learning and deep learning techniques on sensor data.

## Keywords

Deep learning, machine learning, scale-space, mobile computing, Dynamic time warping

## Acronyms

DNN- Deep Neural Networks, DTW- Dynamic Time Warping, DoG – Difference of Gaussians

# INTRODUCTION

This project was developed by Team 17 for the CSE 535 course project. It is a group project and was developed by team of 3 students. The project was originally created and implemented by iMPACT lab and the overall algorithm and methodology was available to us for use. The main goal was to convert the matlab source code into python platform for further research. The project is roughly divided into 15 tasks.

iMPACT lab provided the MYO wristband (MYO: <https://www.myo.com/>), which measures Inertial Measurement Unit (IMU) and electromyogram sensor data. The overall implementation had following process.

* Implement scale-space and Difference of Gaussian algorithm to generate scale-spaces
* Perform extrema based action segmentation and separate user eat and non-eat segments
* Implement Dynamic Time Warping(*DTW*) based feature extraction using scale-spaces
* Implement DTW using C++ and OpenMP for parallel execution
* Implement Generalized Deep Neural Network Training Encode and Decode
* Implement User-Specific Deep Neural Network Training Model Encode and Decode
* Implement Interpolation feature

# Project setup

The entire project is developed in python programming language. Several machine learning libraries are used for implementation as listed below.

* numpy
* scipy
* keras
* tensorflow
* matplotlib.pyplot

We tested the demo of the running code in our laptops. But to run the system completely requires good amount of memory and computation power. As the collected training dataset is quite huge and DTW feature matrix creation and deep neural nets take good amount of computation.

# System Implementaion

**Scale-Space and DoG**

The first step in the implementation is to collect the user data. People are asked to record eating actions while wearing the wristband sensor. This people are called ***donors*** and this data collection is done only once. Lab provided data collected from 40 people to us. The next step is to calculate and generate scale-space data. The sensor data is analyzed by performing Gaussian filter and their differences called difference of Gaussian (**DoG**). Sensor data stream is convolved with a Gaussian signal with zero mean and standard deviation σ = 1. This is first scale-space. Subsequently the process is repeated with different values of σ. The first five scale-spaces of each sensor stream is called an **octave**. Since the MYO wristband has total 18 sensor values - Orientation W,X,Y,Z, Accelerometer X,Y,Z, and Gyroscope X,Y,Z). EMG sensor 8 EMG pods value. The scale-space will be calculated for each of them. Each scale-space consists of **3 octaves,** each octave consisting **5 scale-spaces and 4 DoG.** Total 27 values. The detailed approach is discussed in the paper shared by the instructor, so we are not discussing everything here. Basically, each scale-space captures some hidden features or patterns of the signal.

**Extrema Segmentation**

Next step in the system is to perform extrema-based segmentation on the sensor data. Extrema points are local minima and local maxima in the sensor signals. Since the sensor is worn on wrist while eating, it can never be absolutely still. The collected sensor data will have lots of smaller movements and noise along with actual eating actions. Hence, we segment the data by extrema points and label each segment as eat and non-eat for all the donors.

While implementing the segmentation we experimentally found that the accelerometer Y, Z values had very noticeable peaks whenever user performed EAT. So, this sensor value ca be used for segmentation. As the extremas will be quite clear. Also, to further reduce the effect of noise and

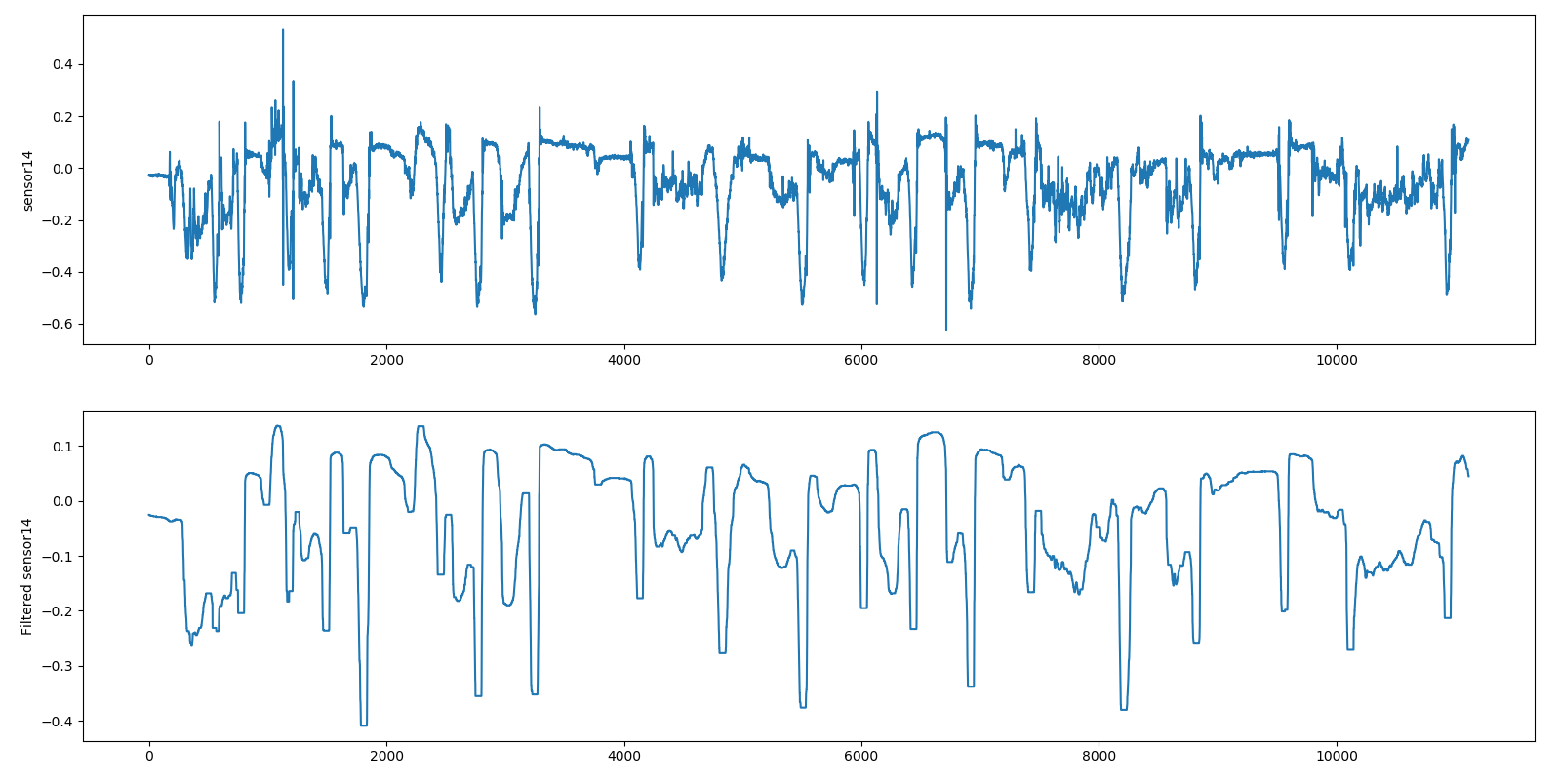


Figure 1: Signal before and after Smoothing. The eat actions are clearly identifiable. [Change the picture with better resolution]

uninterested points in signal we applied simple noise filter to smooth the signal out. It will reduce the no of local minima and maxima points drastically (From 1911 to 100) which reduced the complexity to find and label the eat segments accurately. Locating non-eat segments is a trivial. Once the segment ranges are decided, scale-space computation is performed on them.

Note that each segment is comprised of 18 sensor values. Scale-space is computed for each sensor value. So, there are total 486 values for each donor segment.

18 sensors \* ( 3 octaves \* (5 scale-space + 4 DoG)) = 486.

**DTW Feature Matrix**

DTW otherwise known as Dynamic Time Warping, is the method used for analyzing the above sensor signals. The principle of DTW is to compare temporal sequences in order to determine similarities between these sequences. The idea of this program is to compare a specific scale-space or DoG of a given sensor from a subject segment and compare it to the corresponding scale-space or DoG of the same sensor for the member of the reference subgroup.

In this program, the flow of the program is first to read data from a .csv file and store these values in a data structure. The data structure is grouped based on sensor segment, octave value, and specific scale space or DoG. Each specific scale-space or DoG is normalized and then a DTW algorithm is applied to find the distance between two groups of values. This distance is found for each of the scale-spaces and DoGs of the segment in question (18 sensors \* ( 3 octaves \* (5 scale-space + 4 DoG))), producing 486 distances. These distances are stored in a DTW feature matrix, which is converted to a .csv file for later computation. Each row of this .csv file corresponds to a comparison between two separate segments. Each column is a comparison between a specific scale-space or DoG of the subject segment compared to the corresponding specific scale-space and DoG of the reference segment. Remember that data is segmented based on extrema points, so for each subject and reference sensor signal we define multiple segments.

The technical aspects used here were initially Python libraries, and then Cython librariers. Cython optimizes Python code by using C extensions, and therefore OpenMP was able to be used in order parallelize the DTW feature matrix creation. Task 1.6 required the use of OpenMP which dramatically reduced the runtime of the program since 8 threads were used to create the DTW feature matrix instead of one, as was used in the initial version of the program from task 1.5.

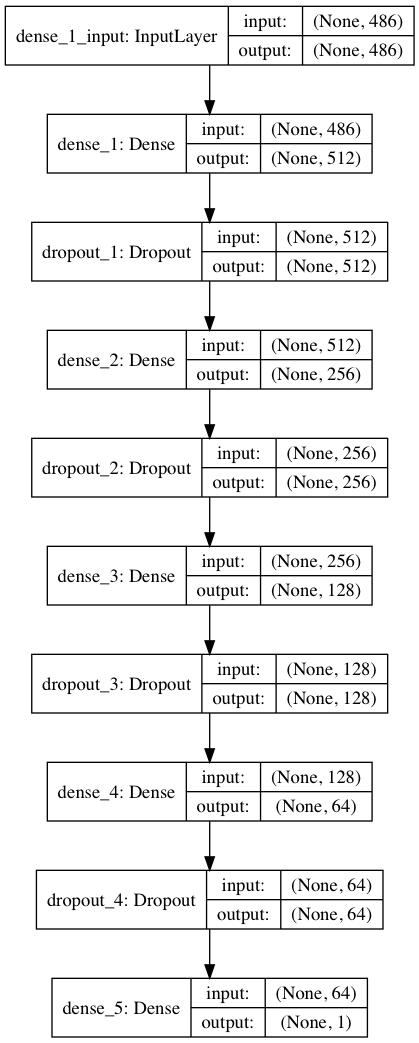
A DTW feature matrix was made for a many-to-many comparison between the donors, which means each segment from each donor was compared to every other donor segment. Video timestamps were used to label these donor segments as eat segments or non-eat segments. Therefore, a DTW feature matrix was generated where 486 columns were created for the scale-space and DoG comparisons, and the 487 column contained a label, where true meant that the segments compared were eat vs. eat and false meant that the segments compared were eat vs. non-eat (non-eat vs non-eat comparisons were thrown out). This particular DTW feature matrix was used for training the General Deep Neural Network.

A DTW feature matrix was also made for a many-to-many comparison between a subject and all the eat segments of the donors. This DTW feature matrix had 486 columns, excluding a label column. This is because the subject did not have labeled segments, since finding the label was the entire purpose of the program. A label is generated based on the steps coinciding with the General Deep Neural Network module and the User-specific Deep Neural Network model.

**Generalized Deep Neural Network model**

As discussed before, inputs to the General Deep Neural Network (DNN) were the many-to-many DTW feature matrix of the donors segments with a label column for training, and a DTW feature matrix for a many-to-many comparison between the subject segments and the eat donor segments for testing. Per the specifications, this General DNN implemented four layers, with nodes exponentially decreasing from 512. The gradient descent optimization is Adaptive Moment Estimation, the activation function is ReLU, and the output layer is sigmoid for binary classification. Figure 3 shows this model representation.

This model was written using Keras. The output of the program was to determine eat segments from the subject, “other” segments from the subject, and generate a most similar users set, meaning those donors that had the most similar segments compared to the user. Each of these similar donors will be used in the next step to generate their own user-specific DNN. The most similar user DTW feature matrix values will be training data for the User-Specific DNN, while the DTW feature matrix of the “other” segments will be the test data. The purpose of this is to better classify the “other” segments as either eat or non-eat.



**Figure 2: Model layout of the General Deep Neural Network**

**User-specific Deep Neural Network model**

After determining the most similar donor set using the generalized DNN model, a donor-specific DNN model is created for every donor in similar user set. We used Keras Python library for the DNN implementation. As discussed in the paper the model is created with 4 hidden layers, each layer having 256 nodes. Input is 486 DTW score values and input layer uses ReLU and output uses sigmoid activation functions. Training the personalized DNN also uses DTW feature matrix calculations. Each donor ‘EAT’ segments are compared with ‘EAT’ segments of all other donors to create EAT x EAT DTW Feature matrix and these entries are labelled as TRUE. Similarly, ‘EAT’ x ‘NON-EAT’ DTW matrix is computed and these entries are labelled false. This labelled data is used for training each individual user-specific DNN. Now this DNN should be able to compare any test segment with that particular donor’s EAT segment and tell how likely it could be a “EAT” segment. Hence, these models are used for identifying “Other” segments of Test user from generalized model as EAT or NON-EAT. Test data for this models are prepared by creating Test user “EAT” vs “Other” segments DTW feature matrix. This data is fed into all the personalized DNN and the output is a weight that describes how many of eat segments of donor matches with one other segment. Hence there will be a vector of weights from all DNN. outputs Now some post processing is applied to interpret these results. If test segment weight is

* 0 from all models, segment becomes NON-EAT
* 1 from all models then EAT
* Exceeds decided threshold for all models- EAT
* “Indefinite” otherwise

All the classified segments can be utilized to classify remaining indefinite segments using the created models.

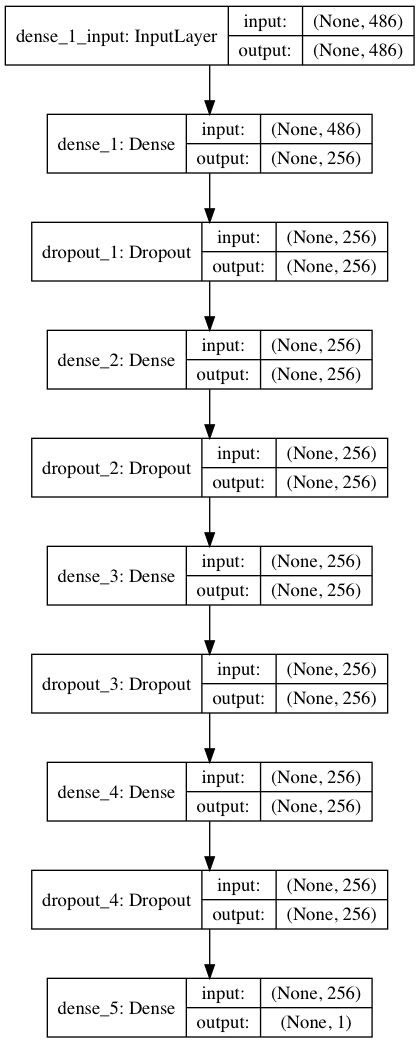


Figure 3: Model layout of the User Specific Deep Neural Network

## TASK DIVISION AND CONTRIBUTIONS

|  |  |
| --- | --- |
| TASK | TEAM MEMBERS |
| 1.1 | Kushal |
| 1.2 | Kushal |
| 1.3 | Kushal |
| 1.4 | Rachel |
| 1.5 | Rachel |
| 1.6 | Rachel |
| 1.7 | Rachel |
| 1.8 | Rachel |
| 1.9 | Kushal |
| 1.10 | Kushal |
| 1.11 | Ashni |
| 1.12 | Ashni |
| 1.13 | Ashni |
| 1.14 | Ashni |
| 1.15 | Ashni |

The entire project was a group effort and most of the tasks are interrelated and dependent on each other. Yet for faster completion and fair division of work we divided 5 tasks per member. The task distribution is shown in the table. But this list is not exhaustive. Certain tasks required great amount of collaboration and collective brainstorming and planning.

# Conclusion

This implementation of the IDEA algorithm was to specification of the paper and proved to work well. The data gathering for detecting whether a subject is eat or not eating is becoming more relevant with the increase in wearables within the connected world. IDEA shows a simple example of gesture recognition and the potential for action detection in the future of wearables; think of the abstractions in the realms of sign language, gaming, sports, therapy, and more! Important future work will benefit from programmatic examples as demonstrated in this paper.

# ACKNOWLEDGMENTS

We sincerely appreciate the efforts of Mr. Junghyo Lee, TA of this course for providing all the guidance to carryout this project. We would like to thank the instructor Dr. Ayan Banerjee for making the course very intriguing, thought provoking and a great learning opportunity.

# REFERENCES

1. <https://www.youtube.com/watch?v=NPcMS49V5hg>, Scale-invariant Feature Transform (SIFT),
2. https://www.olympus-lifescience.com/en/microscope-resource/primer/java/digitalimaging/processing/diffgaussians
3. The unpublished IDEA paper shared with us