Hidden Markov Models for Gesture Recognition

Rachit Bhargava and Nikolay A. Atanasov

***Abstract*— This work addresses the task of classifying a gesture from raw IMU data using the Hidden Markov Model. A Hidden Markov model is used because the system is partially observable and the state of the robot is 'hidden'. The only information provided is the raw IMU data from the mobile phone.**

I. INTRODUCTION

This project discusses the classification of a gesture using data from an IMU. The IMU was used to trace the motion of the user. The 6 classes considered were beat3, beat4, circle, eight, infinity and wave. A Hidden Markov model (HMM) was trained using this data. The parameters are estimated using an Expectation Maximization based algorithm known as the Baum Welch algorithm. The HMM consists of 3 parameters, namely the Transition matrix (T) which encodes the probability of going to the any other state (including the same state) given the previous state, the Emission matrix (B) which encodes the probability of making a particular observation given the state and the prior probability on the initial state (π).

The following section discusses the data and the clustering process. The following sections discuss the training procedure and the testing procedure. Finally, the results are presented along with a discussion of the results attained.

II. DATA AND K-MEANS CLUSTERING

This section discusses the data that was given and the K-Means clustering done on the data.

A. Data

Data was attained from the IMU of the mobile phone used. It was used to capture the features of the gesture made by the motion of the phone. The IMU returned a 6 dimensional vector with the first 3 elements containing the accelerometer data and the last 3 elements containing the gyroscope data. The data was used in its raw form. However, experiments were done by taking lower dimensional variants of this 6D vector. A 5 dimensional vector was taken in which the first element is the norm of the x and y components of the accelerometer, the second element encoded the direction (estimated using Atan2) and the last 3 elements were the raw gyroscope data. However, it reduced the in training accuracy. So the original 6D vector was used to avoid wrong classification due to loss of information by taking a lower dimension vector.

*This work is a project as part of ESE 650 Learning in Robotics at the University of Pennsylvania

B. K-Means clustering

An important step before training the Hidden Markov Model for this project is to discretize the continuous observation space. One way to do this is by clustering the data and representing each data point by the centroid of the cluster it belongs to. The entire training data (original and additional) was compiled into a single array and passed to the K-Means clustering method present in SciKitLearn. The number of clusters is represented by M.

III. TRAINING THE HMM

This section describes the initialization of the HMM parameters, the E step and the M step of the training process. Note that the various parameters $\alpha_t^k, \beta_t^k, \gamma_t^k, \chi_t^k$ are estimated at each time step t for each dataset k in the M step of each loop of EM. Parameters from each dataset are used to update the parameters T, B, π of the HMM model for the class in the M step.

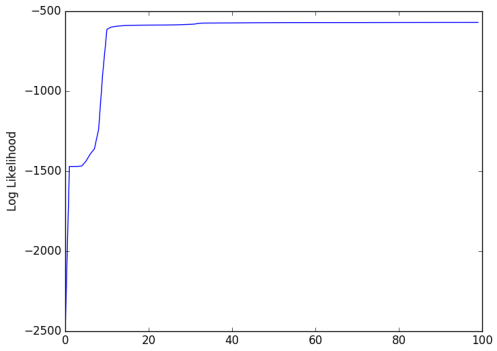
Before the loop of EM begins, each dataset corresponding to a particular class is collected. The centroid it belongs to is t Following that the model parameters are initialized.

A. Initialization

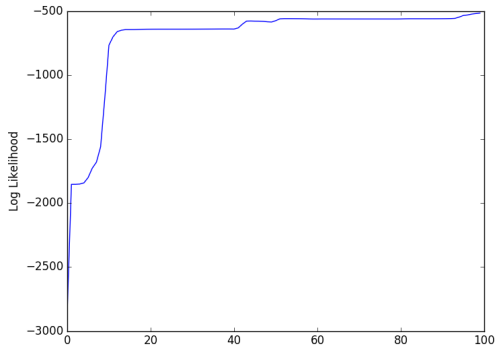
There are 2 parameters of the HMM model parameters depend, namely the number of discrete states N and the number of centroids for clustering the observation data, M. For the currently trained model, N = 15 and M = 30. Ideally these parameters must be decided upon using cross validation techniques like 10 fold cross validation or Leave One Out Cross Validation. But due to time constraints, the parameters were decided upon by training on the original training dataset and testing on the additional training dataset. After deciding N and M, the 3 model parameters are initialized in the following manner:

T: A left to right model was assumed for the HMM model. This implies, that an assumption was made initially, that the state could only be updated to its own state or the next state. Further an assumption was made that the probability of going to either state (the same or the next) was the same. The T matrix has the shape of N x N. So the T matrix initialization looked like the following:

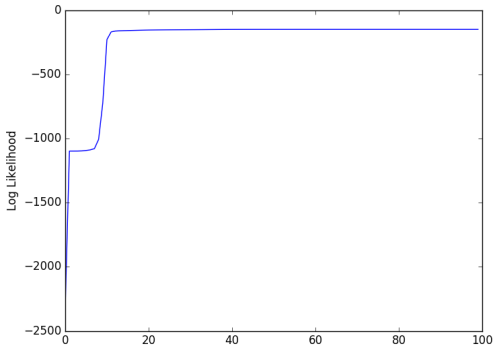
$$\begin{bmatrix} 0.5 & 0 & \dots & 0 & 0.5 \\ 0.5 & 0.5 & & & 0 \\ 0 & 0.5 & 0.5 & & \vdots \\ \vdots & & & \ddots & 0 \\ 0 & 0 & \dots & 0 & 0.5 \end{bmatrix} \tag{1}$$



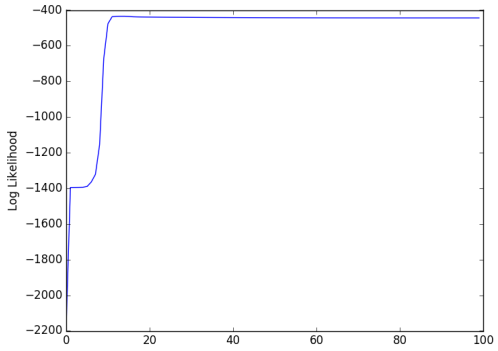
(a) Beat3



(b) Beat4



(c) Circle



(d) Eight

Fig. 1: Log likelihood vs iterations for various classes

B: An assumption was made that each observation is equally likely to be observed at any state. The B matrix, emission matrix has the shape M x N and its initialization looks like the following:

$$\begin{bmatrix} 0.5 & 0.5 & \dots & 0.5 \\ 0.5 & 0.5 & & 0.5 \\ \vdots & & \ddots & \vdots \\ 0.5 & 0.5 & \dots & 0.5 \end{bmatrix} \tag{2}$$

π**:** Due to the assumption of a left to right model, the motion is assumed to begin explicitly from state one and no other. Hence the initialization of π of size (1 x N) attains the following form:

$$\begin{bmatrix} 1 & 0 & \dots & 0 & 0 \end{bmatrix} \tag{3}$$

B. M-Step

Once the model parameters are initialized, the EM loop begins. The M-step of the Baum Welch algorithm involves estimating the parameters $\alpha_t^k, \beta_t^k, \gamma_t^k, \chi_t^k$ are estimated at each time step t for each dataset of the class k. This is done using the current estimates of the model parameters T, B, π.

The α parameters are estimated using the Forward algorithm [1]. The β parameters are estimated using the Backward algorithm [1]. The additional parameters γ, χ are estimated using the formulas specified in [1].

A big problem during implementation of this algorithm was the problem of underflow. Sometimes the values are estimated so small that the float variable sets it to 0. This might result in division by zero errors during estimation of γ, χ. This is solved by scaling the parameters as described in [2].

C. E-step

The estimation step involves updating the HMM model parameters. This was done using the parameters $\alpha_t^k, \beta_t^k, \gamma_t^k, \chi_t^k$ estimated for each dataset for each time step for a particular class and the equations mentioned in the slides [1].

D. Termination of EM Loop

The EM loop terminates either when the maximum number of iterations are crossed (100 in this project) or the log likelihood converges ie the difference between the log likelihood at the previous and current time step is below a particular threshold. The plot of the log likelihood for 4 of the classes are shown in Fig. 1.

The cumulative log likelihood is estimated using the scaling parameters, c_t (extracted in the M step) as using the equations derived in [2].

$$P(Class|Parameters) = -\sum_t \log(c_t) \tag{4}$$

IV. Testing

Once the training is done, the model parameters T, B, π are attained for each class. Testing begins by classifying each row of the input dataset to a particular centroid using the KMeans model estimated in section II. This is the observation vector.

Using the parameters from each class and the observation, the scaling parameters are extracted by the implementing the forward algorithm as mentioned in the previous sections. Using the scaling parameters, the cumulative log likelihood is estimated using equation (4) for each class. The input dataset is assigned the class which produces the maximum likelihood.

The results from the given test set are discussed in the next section.

V. Results

The results from implementation of the algorithm on the 'single' and 'multiple' sets are shown in Fig. 2 and 3 respectively. The caption of each figure exhibits the final classified gesture which is the gesture with the maximum confidence value. The plots signify the gestures and the confidence of classification associated to each of them. For most of the cases the maximum confidence value is relatively high is pretty high with respect to the remaining confidence values. But for cases of beat4 and beat3 classes, there occurs a lot of confusion in classification ie the confidence values are quite comparable as can be seen in Fig 3l. This implies that the gestures beat3 and beat4 are quite similar.

There is still room for improvement in the algorithm. As mentioned before it is essential that time is spent to tune the parameters carefully using Cross Validation. Alternative techniques of initializing the model parameters of the HMM can also be explored to improve the results of the HMM.

VI. CONCLUSIONS

In this paper, an algorithm to classify a gesture using a Hidden Markov model is discussed. The HMM was trained on raw IMU data extracted from a mobile that the user holds while making the gesture.

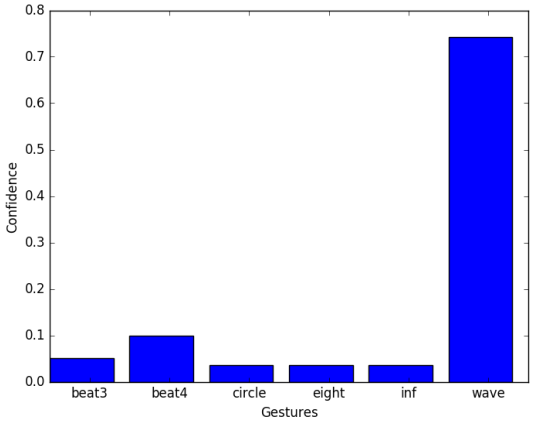
This project helped develop a very good understanding about how to deal with situations where the model data is partially observable (in this case the states were hidden) and no information of the control action is known. It was really interesting to be able to facilitate the classification using a Markov chain and observations at each time step.

ACKNOWLEDGMENT

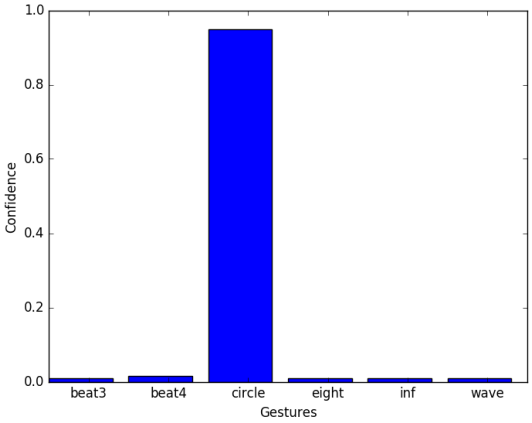
I want to thank Dr. Atanasov and the TAs for helping me with my doubts on Piazza.

References

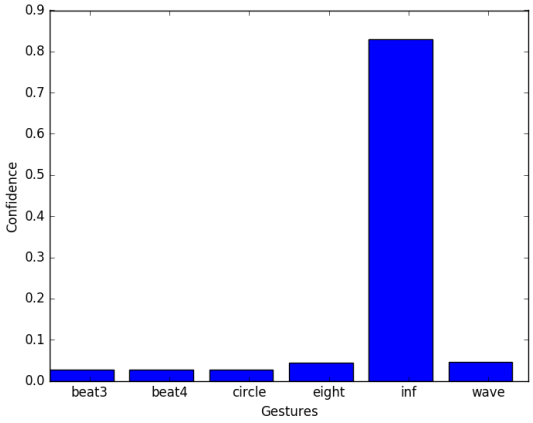
[1] Slides for ESE 650 from Dr. Atanasov
[2] A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, Lawrence R. Rabiner



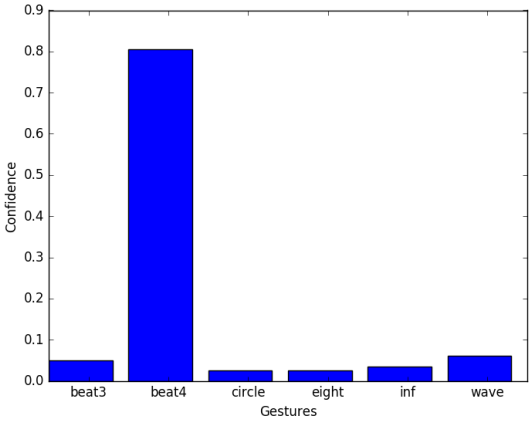
(a) Result: Wave



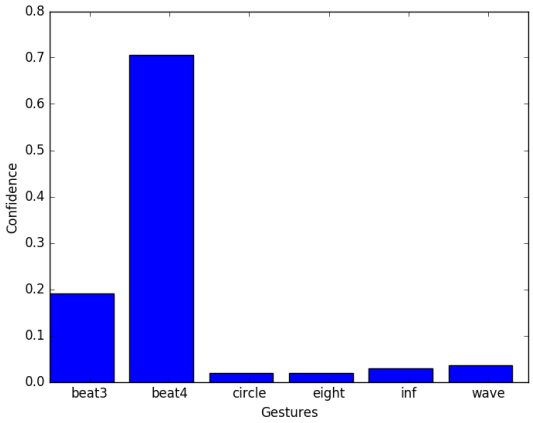
(b) Result: Circle



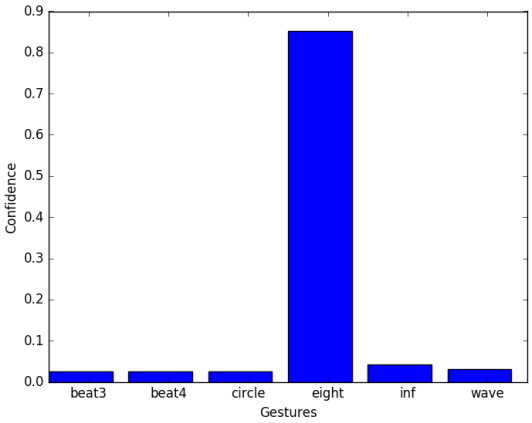
(c) Result: Infinity



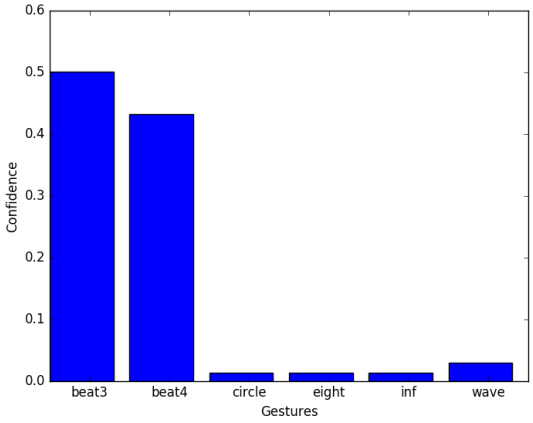
(d) Result: Beat4



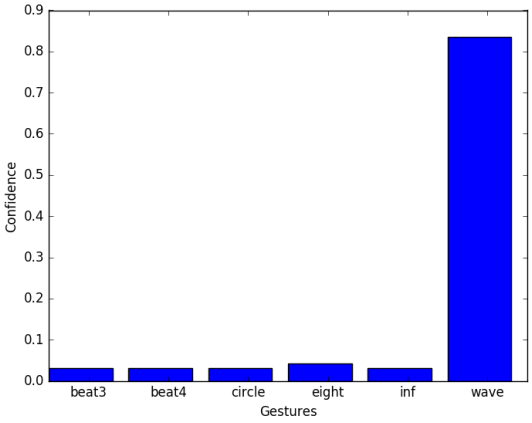
(e) Result: Beat4



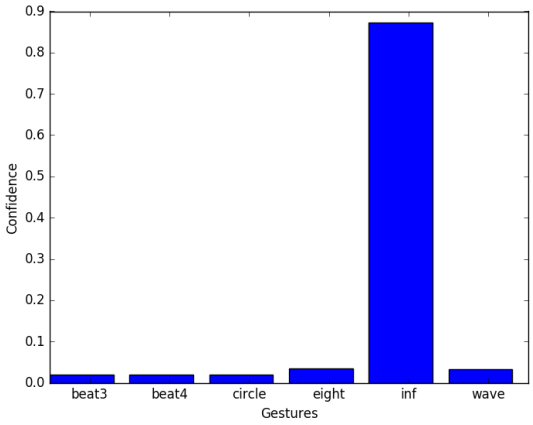
(f) Result: Eight



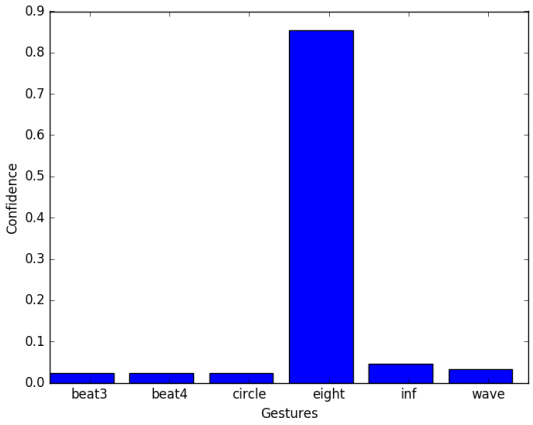
(g) Result: Beat3 (Choice 1) or Beat4(Close 2nd)



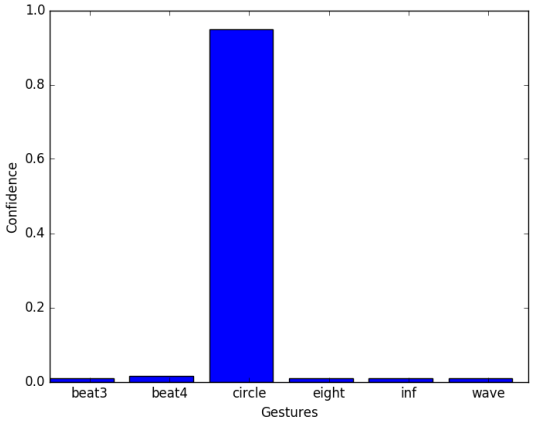
(h) Result: Wave



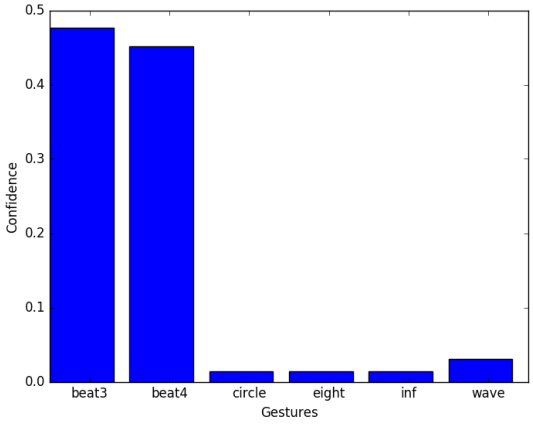
(i) Result: Infinity



(j) Result: Eight

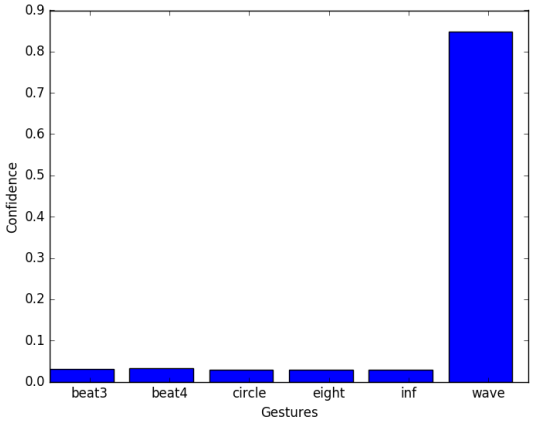


(k) Result: Circle

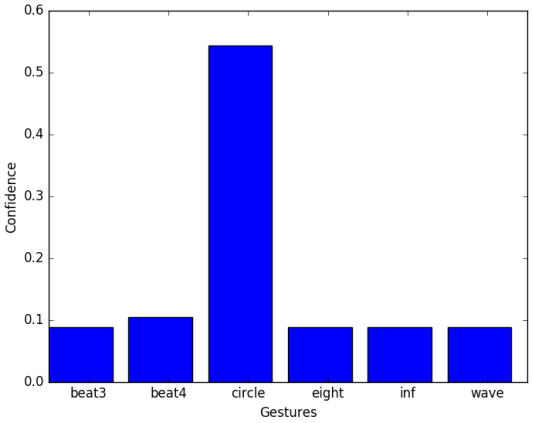


(l) Result: Beat3(choice 1) or Beat4(close 2nd)

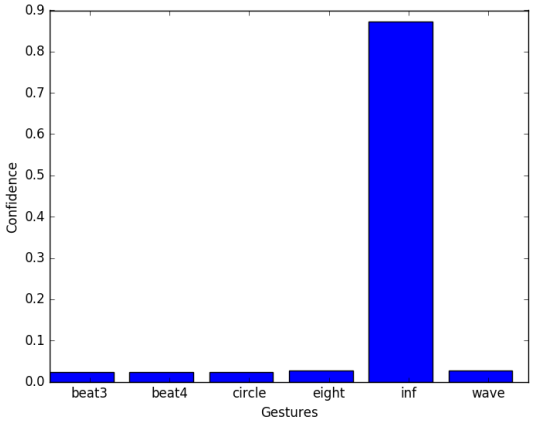
Fig. 2: Results of implementation of the algorithm on 'single test set'



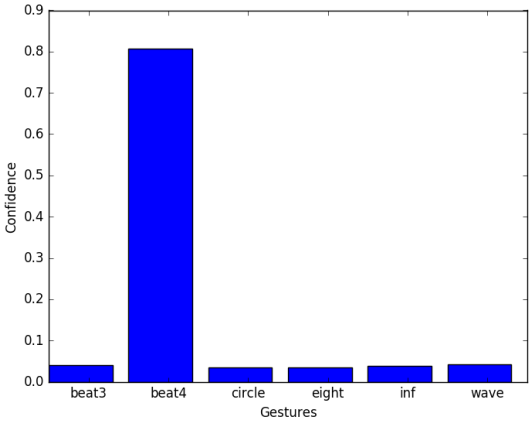
(a) Result: Wave



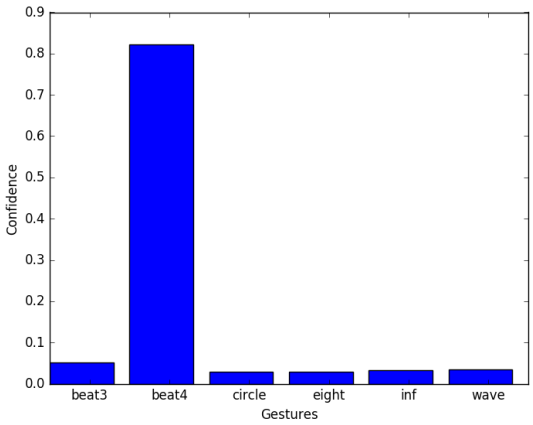
(b) Result: Circle



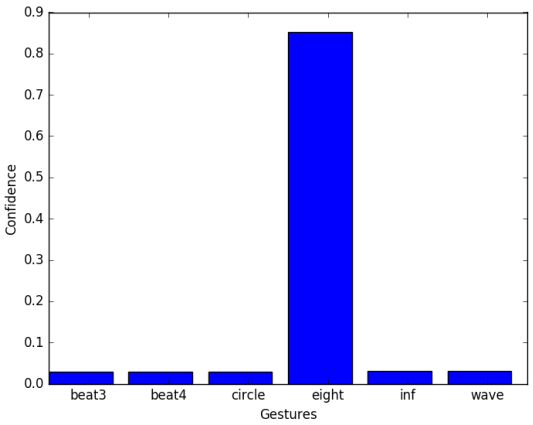
(c) Result: Infinity



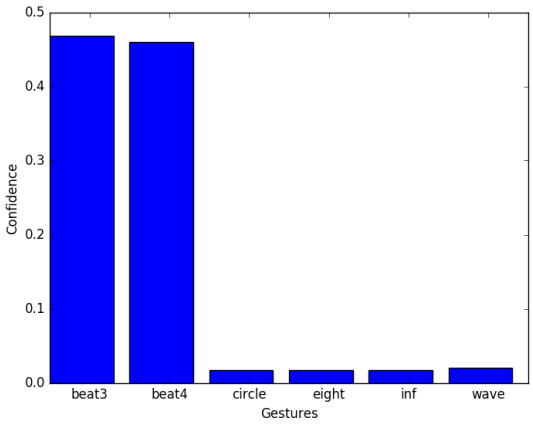
(d) Result: Beat4



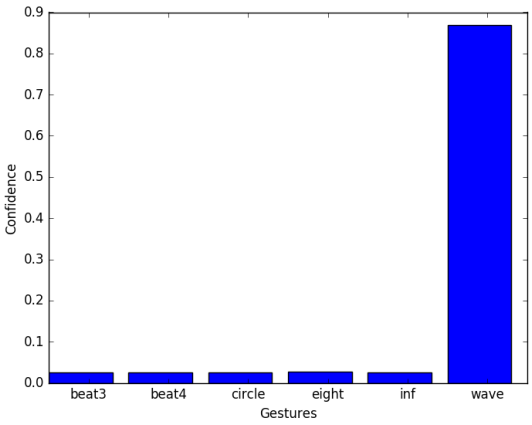
(e) Result: Beat4



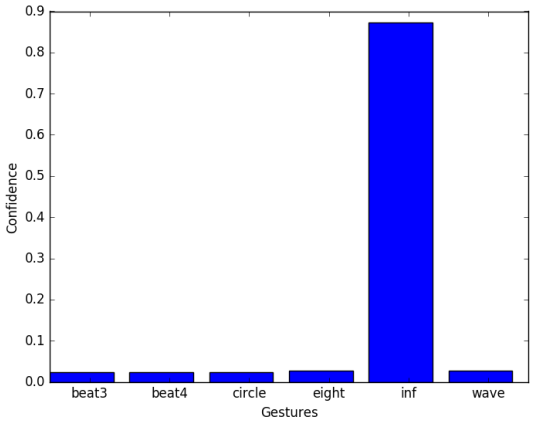
(f) Result: Eight



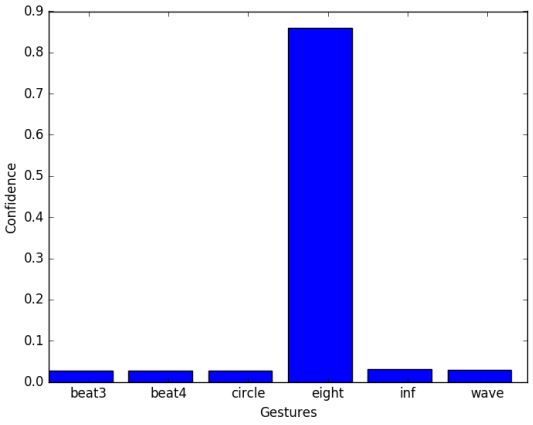
(g) Result: Beat3 (Choice 1) or Beat4(Close 2nd)



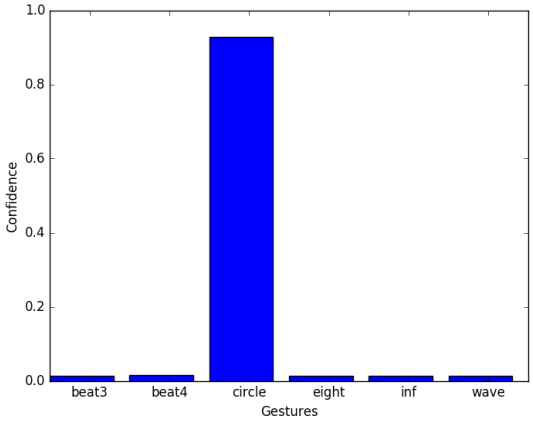
(h) Result: Wave



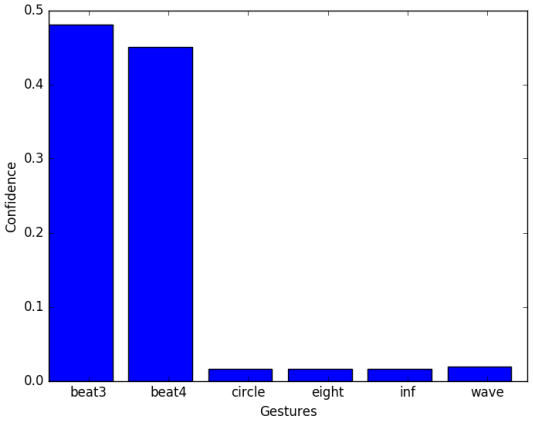
(i) Result: Infinity



(j) Result: Eight



(k) Result: Circle



(l) Result: Beat3(choice 1) or Beat4(close 2nd)

Fig. 3: Results of implementation of the algorithm on 'multiple test set'