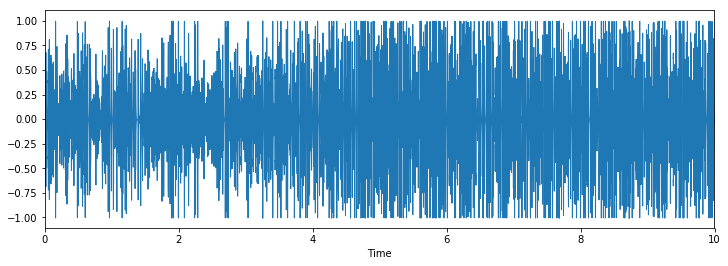
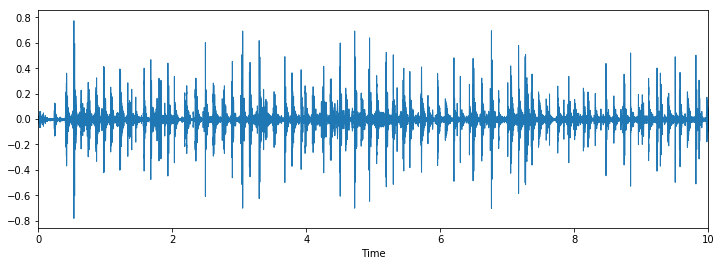
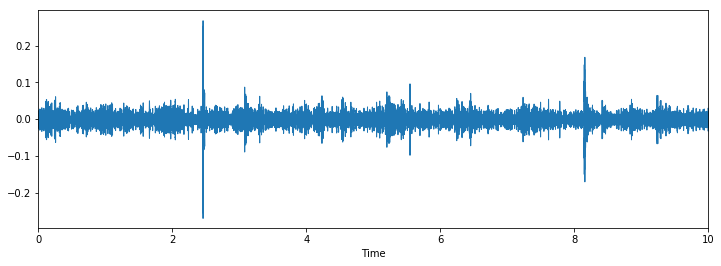
Data Collection and Visualization:

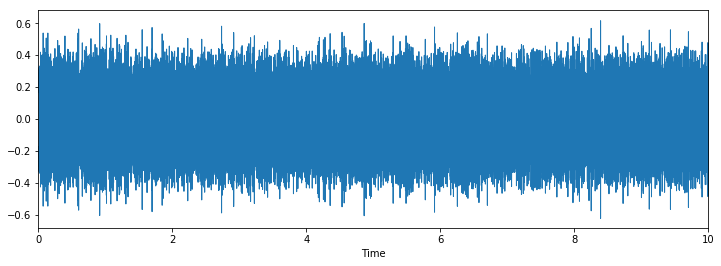
* We collected data for all 5 activities – Laundry, Eating, Hair drying, Vacuuming, Typing using the USB Microphone interfaced with the Raspberry pi
* We collected 50 audio samples of each activity for a duration of 10s. Below table represents the various type of data collected for each activity.

|  |  |
| --- | --- |
| ACTIVITY | DATA COLLECTED |
| Hair Drying | Data was collected on all 3 speed and heat settings |
| Laundry | Data was collected on washer and Dryer, when they were running on varying speeds. (i.e – when the machine switched on vs when the machine is running at full speed) |
| Eating | Data was collected on all kinds of food- soft and hard. Particularly we collected data when the subject was eating Chips, Bread, Apple, Grapes, Chocolate, Cookie, Chewing Gum. |
| Vacuuming | Data was collected on all setting on the vacuum cleaner- low to high. |
| Typing | Data was collected when the subject was typing rigorously, when the subject was typing slowly and when the subject is typing and using mouse simultaneously. |

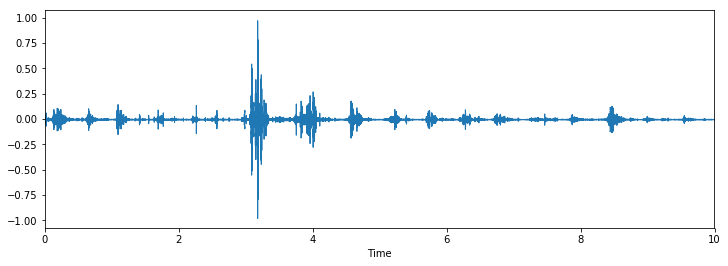
* We then visualized data by plotting them using Python to see if they were actually different and if we can run a classifier to classify them accurately. Below are the plots for each activity.
  + HairDrying
  + Typing



* + Laundry
  + Vacuuming



* + Eating



It can be seen from the above plots that the audio signals have different principle components and can be classified with a classifier.

Feature extraction and building Feature Vector:

For the feature, since audio was being used we decided to use a feature in the frequency domain. Based on literature we found that for both audio and speech recognition MFCC or mel-frequency cepstral coefficients were extensively used. This works on the concept of human hearing , which can differentiate better in the mel frequency scale rather than the normal frequency scale. Each sound has a particular shape and this shape can be accurately described as an envelope of a short time power spectrum which is given by the MFCCs. They are generally used for Automatic Speech recognition , but however they can be used for other sounds as well.

The MFCCs are calculated as follows:

1. Divide the audio into several frames (0.25 seconds with overlap of 0.1s)

2.Calculate the periodogram estimate of the power spectrum

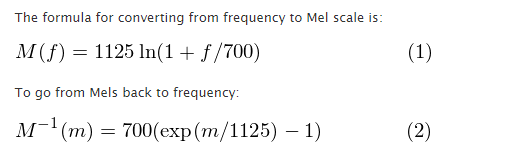
3. Apply the mel filter banks by taking product of both the energy and the mel banks

4. Taking logarithm of the entire value and computing Discrete Cosine Transform (DCT)

5. Taking the first 13 features, and discarding the rest

Apart from the MFCC to improve on the accuracy delta and d-delta components are used. Delta is the differential of the first frame of MFCC with the second frame and so on. D-delta is the difference of the delta components.

The audio is divided into several frames as the audio tends to kind of stationary in small time scale and hence this property can be exploited by dividing into frames. The power spectra is calculated as the human ear (cochlea ) vibrates differently at different powers based on the frequency and this can be used to differentiate the sound. However they cannot differentiate between minute differences in frequencies or sounds for which the the mel banks are used. The power periodogram is put in different bins and summed together. At higher frequencies the variations of energy are not important and hence the mel scale is used which provides in proper spacing of the bins.



The mel banks are computed as follows:

1. Choose lower frequency as 0Hz and upper frequency as 24Khz (fs/2)

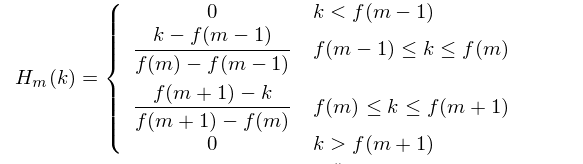
2. Convert them to the mel frequency scale

3. Choose 20 evenly spaced points between them for the 20 filter bank values

4. Convert them back to the frequency scale (h(i))

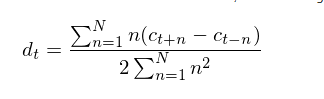
5. Find the corresponding FFT bins for the frequency resolution as f(i)= (fftno\*h(i)) /samplerate- here the fft number is the fft taken during the calculation of the power which is calculated using the sample rate and window length. Here fftno=2048.

6. Obtain the mel banks using the following equations,



Find the product of the mel banks and the power values and take the logarithm as the human ear hears better in the log scale compared to the linear scale. DCT is taken as the audio frames are overlapping and the discrete cosine transforms helps in exploiting the property of correlation. Majority of the data or the information is stored in the first 13 values of the MFCC coefficients and hence the rest are discarded.

Delta and d-delta components give the variation of the MFCc and the delta coefficients respectively. They are obtained the padding the MFCC with 2 zeros at the beginning and at the end and finding the dot product of 5 components and finding the sum across all . This is given by the formula below



Here c=MFCC, N=2, ( t+n to t-n corresponds to 5 values being taken for computation). Applying this formula for the delta coefficient, gives the d-delta coefficients.

In this project, we took 10s audio which gave us 1000 frames with each having 13 MFCC,delta and d-delta coefficients. They were all combined together to give a feature vector of size 39000. To train the models , 50 data samples of each activity were taken.

Machine Learning Algorithms:

*Non- Real Time-based prediction:*

* Now that we have the feature vector, we split the data in ratio of 75:25 for train and test respectively.
* We first implemented the Non-Real Time version of the prediction – i.e. We recorded the entire audio first, extracted the feature vector and ran the ML algorithm to predict the activity. The reason this is not real time is because it does not predict as the activity comes in.
* Since we are running the algorithms on Raspberry Pi, we wanted to use the classification algorithm that gives us the maximum accuracy and with least execution time.
* In order to find this out we ran all the classification algorithms and noted their execution time and accuracy and took an average of 5 runs for different combinations of the three key features – MFCC, Delta, DDelta.
* The below table shows the results:

**Feature: MFCC**

|  |  |  |
| --- | --- | --- |
| ML Algorithm | Accuracy | Execution Time (in seconds) |
| Logistic Regression | 73.01% | 95.17 |
| SVM – Linear Kernel | 66.67% | 7.58 |
| SVM -RBF Kernel | 66.67% | 110 |
| K- Nearest Neighbors | 79.36% | 3.192 |
| Decision Tree | 100% | 7.6 |
| Multilayer Perceptron Classification | 47% | 23.28 |
| Random Forest | 100% | 0.6 |
| Boosting- Ada Boost | 99.92% | 185 |

**Feature: MFCC and Delta**

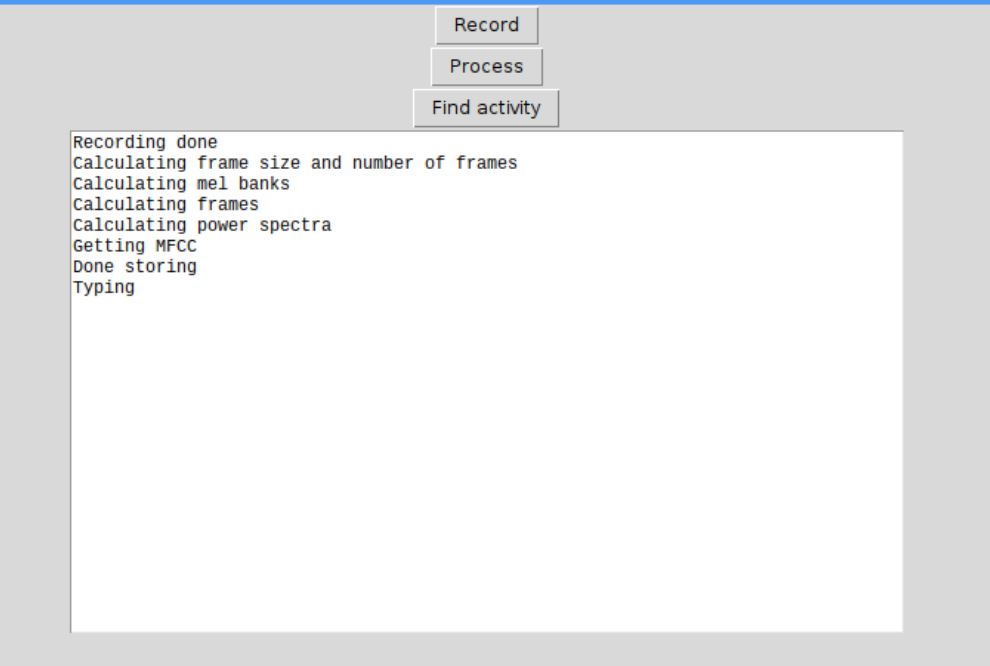
|  |  |  |
| --- | --- | --- |
| ML Algorithm | Accuracy | Execution Time (in seconds) |
| Logistic Regression | 87.3% | 218 |
| SVM – Linear Kernel | 77.77% | 235 |
| SVM -RBF Kernel | 84.12% | 14.7 |
| K- Nearest Neighbors | 77.77% | 6.4 |
| Decision Tree | 100% | 13.7 |
| Multilayer Perceptron Classification | 42% | 46.53 |
| Random Forest | 100% | 0.95 |
| Boosting- Ada Boost | 99.92% | 365.9 |

**Feature: MFCC, Delta and DDelta**

|  |  |  |
| --- | --- | --- |
| ML Algorithm | Accuracy | Execution Time (in seconds) |
| Logistic Regression | 94.1% | 325 |
| SVM – Linear Kernel | 88.1% | 235 |
| SVM -RBF Kernel | 84.12 | 14.7 |
| K- Nearest Neighbors | 77.77 | 6.4 |
| Decision Tree | 100% | 13.7 |
| Multilayer Perceptron Classification | 42 | 46.53 |
| Random Forest | 100% | 0.95 |
| Boosting- Ada Boost | 99.92% | 365.9 |

From the table, Random forest with one feature gives the best accuracy and execution time. We implemented the Non-real time-based activity prediction with one feature -MFCC and with Random Forest as our classifier algorithm.

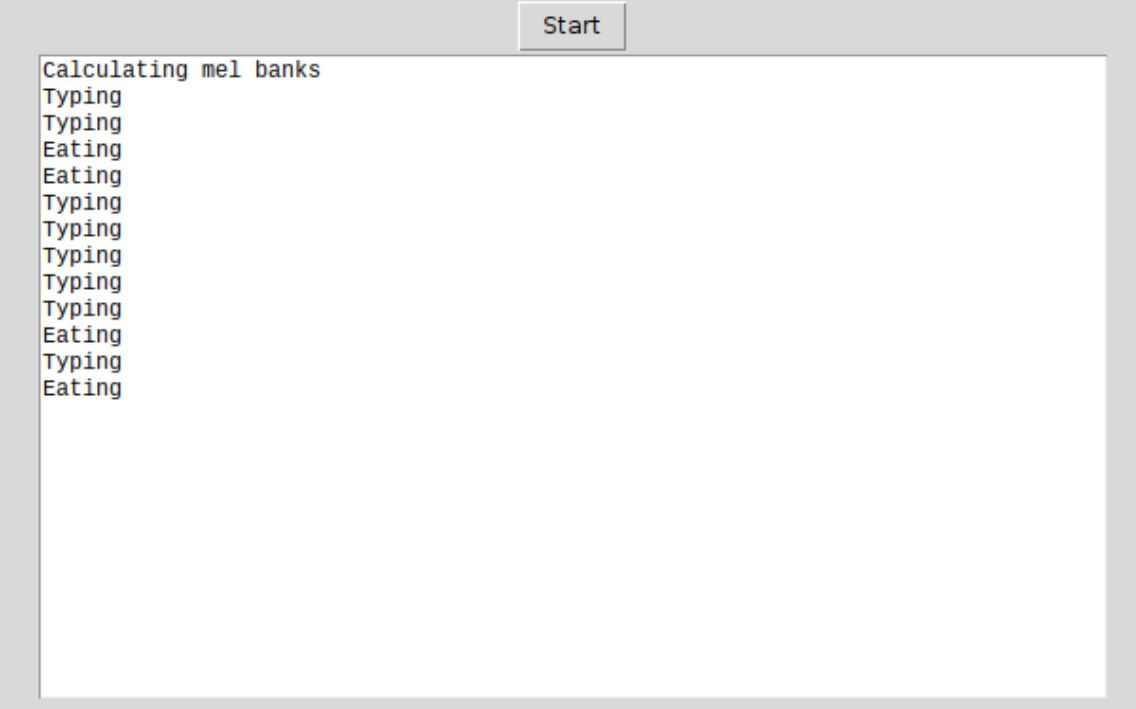
* All the above steps were done in Jupyter Notebook and iPython Notebooks. To make the project more interactive, we developed a GUI using Python’s Tkinter library.
* The GUI has three buttons for the following purposes:
  + Record – signals the Microphone interface to start recording for a duration of 10sec
  + Process – Opens the recorded audio file, builds MFCC feature vector by following the steps in Section 7 and stores the feature vector in a text file.
  + Find activity – Opens the text file, runs the Random forest algorithm to predict the label. Based on the label, it prints out the type of activity that was recorded.
* Below is a picture of the GUI detecting Typing



* During demo, we were not able to show the working of activities such as Laundry and Vacuum. Hence, we recorded a video and below are the links:
  + Laundry- <https://youtu.be/0m3kBEFjWR4>
  + Vacuum- <https://youtu.be/f9YrJ3E30Mk>

*Real Time-based Prediction*:

* We implemented the real-time based prediction using multithreading by running two threads in parallel, one thread to record audio and one thread to run the Machine algorithm to predict the activity.
* We have a common queue which is shared between the two threads. The data that is being recorded is put in the queue by the Thread\_One. Thread\_Two consumes the data in the queue builds the MFCC feature vector.
* The built feature vector is then used to predict the activity by running Random Forest algorithm.
* Our main issue with this approach was that the algorithm was not predicting correctly and latency. The audio is 10s long and it is simultaneously calculating the feature vector as the audio comes in, because of this there was latency involved.
* To increase the probability of predicting correctly, we considered three features (MFCC, Delta, DDelta) for the real-time implementation as it will have more principle components for each label. This improved the prediction and our model was able to correctly classify the activities, but the latency issue persisted.
* All the above steps were done using Threading library in Python using Jupyter’s Ipython notebook.
* The GUI has only one button:
  + Start – Starts the process by executing Thread\_One and Thread\_Two
* Below is a picture of GUI for a series of activities- Typing and Eating



* During demo, we were not able to show the working of activities such as Laundry and Vacuum. Hence, we recorded a video and below are the links:
  + Laundry- <https://youtu.be/Ud1WwnJU2n0>
  + Vacuum- <https://youtu.be/93BD09TxFtk>

*Issues Faced:*

* The main issue we had was in reducing the latency in real-time implementation.
* We implemented an approach of having a sliding window kind which samples signals in blocks- i.e for a 9s audio signal, it samples first 3 seconds and builds the feature vector, then samples the next 3 seconds but it retains 1 second of the last frame and gets 2 seconds of the new frame in order to retain continuity.
* The above approach did not increase or decrease the latency , but it greatly affected the prediction as none of the activities were predicted correctly.