Question 1:

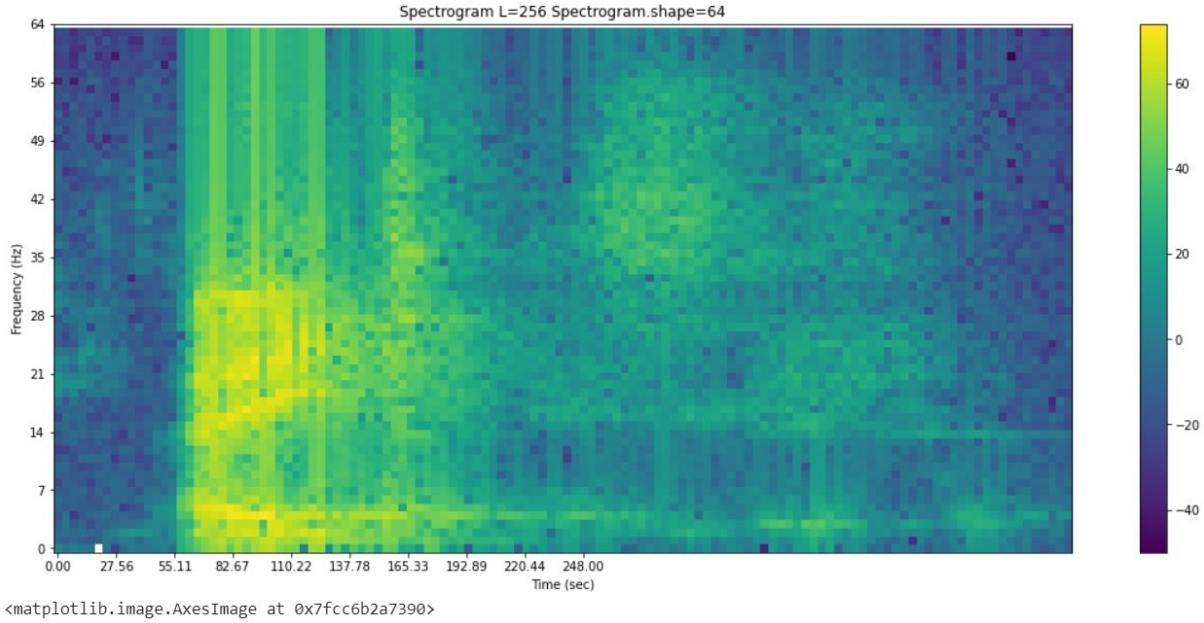
All the features and models are saved in the pickle files uploaded with the code.
Constraints:

1. DFT is used in place of FFT as in FFT we the array length should be a power of 2. Which was causing some problem in the dimensionality.
2. Window size= 128
3. Stride size 64
4. Nylimit = len(audio/2)

Implementation:

1. We divide the audio in different windows and then traverse through these windows with particular stride size. We calculate their DFT. The array with these DFT give us the spectrogram

Plots:



<matplotlib.image.AxesImage at 0x7fcc6b2a7390>

This is the plot of the spectrogram of the sound of 8

X_train initial - `/content/drive/My Drive/Pickle_X_train2`
Y_train - `/content/drive/My Drive/Pickle_y_train2`
Model - `/content/drive/My Drive/Pickle_Model_ques1_2`
X_test initial - `/content/drive/My Drive/Pickle_X_test2`
Y_test - `/content/drive/My Drive/Pickle_y_test2`
X_train_final - `/content/drive/My Drive/Pickle_X_train_final2`

References:
https://towardsdatascience.com/fast-fourier-transform-937926e591cb

https://fairyonice.github.io/implement-the-spectrogram-from-scratch-in-python.html
https://towardsdatascience.com/understanding-audio-data-fourier-transform-fft-spectrogram-and-speech-recognition-a4072d228520
https://timsainburg.com/python-mel-compression-inversion.html
https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python

Question 2:

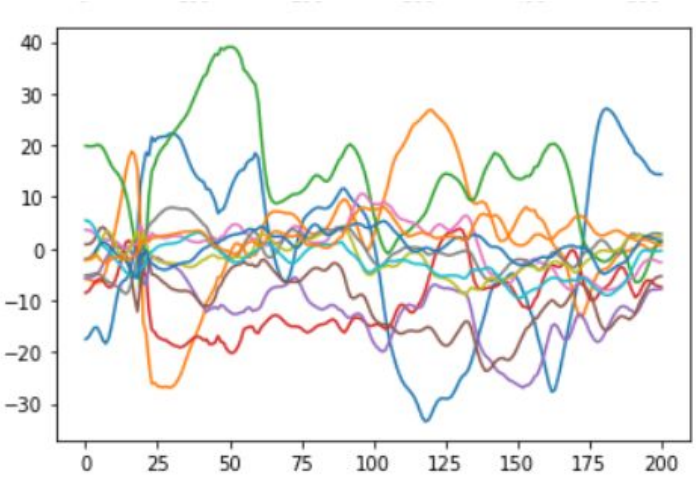
Constraints:

1. Fourier size = 1024
2. Stride_size = 5
3. Number of mel filters = 13
4. Norm = ‘ortho’

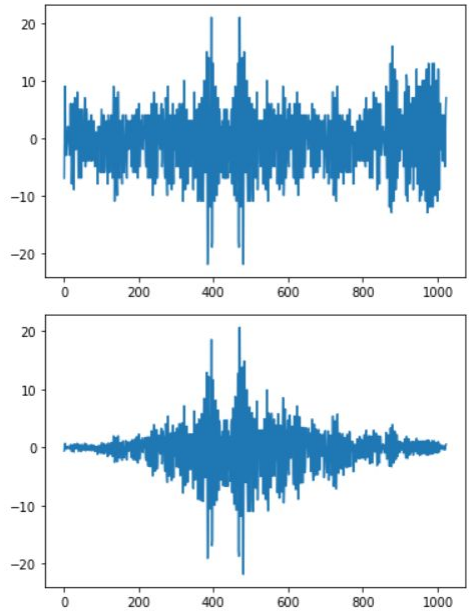
Implementation:

1. First we divide the audio in different frames
2. Hamming the audio in different frames so that the start and the end can be close to 0
3. Taking FFT of each frame
4. Getting the filter points and frequencies in mel form
5. Making of filters
6. Applying mel filterbank to power spectra
7. Applying DCT

Plots:



MFCC Plot



Audio after diving in filters and hamming it.

X_train_inintial - `/content/drive/My Drive/Pickle_MFCC_X_train2`
Y_train - `/content/drive/My Drive/Pickle_MFCC_y_train2`
Model - `/content/drive/My Drive/Pickle_MFCC_Model_ques2_2`
X_train_final - `/content/drive/My Drive/Pickle_MFCC_Model_ques2_2`
X_test_initial - `/content/drive/My Drive/Pickle_MFCC_X_test2'`
Y_test - `/content/drive/My Drive/Pickle_MFCC_y_test2'`

References:

https://www.kaggle.com/ilyamich/mfcc-implementation-and-tutorial

https://github.com/amanbasu/speech-emotion-recognition/blob/master/mel.ipynb

http://www.cs.cmu.edu/~dhuggins/Projects/pyphone/sphinx/mfcc.py

http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/

Question 3:

Spectrogram Results:
We have used the polynomial kernel of degree 3 for the SVM
Accuracy: 0.5210843373493976

Average = None

Precision: [0.53584906 0.77868852 0.80124224 0.66887417 0.52222222 0.72928177
 0.84699454 0.5739645 0.65217391 0.22739018]
Recall: [0.61471861 0.4185022 0.49236641 0.46759259 0.43925234 0.54771784
 0.63265306 0.4254386 0.41666667 0.72131148]

Average = ‘Weighted’
Precision: 0.6356232838641535
Recall: 0.5210843373493976

Average = ‘Macro’
Precision: 0.6336681106064812
Recall: 0.517621980073259

Class 6 has the best precision
Class 0 has the best recall

MFCC Results:
We have use the rbf kernel for the SVM
Accuracy: 0.7971130713712911

Average = ‘Macro’
Precision: 0.7958552545790711
Recall: 0.7885062490435224

Average = ‘Micro’
Precision: 0.7902967121090617
Recall: 0.7902967121090617

Average = None
Precision: [0.80542986 0.80861244 0.86397059 0.63508772 0.62416107 0.8442623
 0.92207792 0.81702128 0.74358974 0.89433962]
Recall: [0.73251029 0.69834711 0.83928571 0.78695652 0.80869565 0.78326996
 0.8129771 0.77419355 0.73728814 0.91153846]

Average = ‘Weighted’
Precision: 0.8008038539795207
Recall: 0.7902967121090617

Class 6 has the best precision
Class 0 has the best recall

The model using the MFCC features performs better than the model using the Spectrogram features. It has a higher accuracy, Recall and Precision than the spectrogram model.

This is because:

1. MFCC uses the MEL filter power banks to get the power spectra.
2. We use Discrete Cosine Transform (DCT) in the MFCC instead of inverse fourier transform. The advantage that it has is that it returns real values and also has more efficient computation and storage.
3. The intervals in spectrogram are equispaced where in MFCC they are not which gives better results.

Adding Noise randomly in our samples and analysing the results.
I have added random noises in the sounds provided.

Spectrogram
Accuracy: 0.3408179631114675

Average = Macro
Precision: 0.34151547482303546
Recall: 0.33861491959355183

Average = 'Weighted'
Precision: 0.34449029979642976
Recall: 0.3408179631114675

Average = None
Precision: [0.26625387 0.42168675 0.44528302 0.33913043 0.2625 0.375
0.35582822 0.27004219 0.25490196 0.4245283]
Recall: [0.35390947 0.4338843 0.42142857 0.33913043 0.27391304 0.29657795
0.44274809 0.25806452 0.22033898 0.34615385]

MFCC

```
Accuracy: 0.4294306335204491

Average= 'Weighted'
Precision: 0.504707368552182
Recall: 0.4294306335204491

Average = None
Precision: [0.33684211 0.60209424 0.52988048 0.24006623 0.41125541 0.4
```

```
 0.75       0.50251256 0.54032258 0.68911917]
Recall: [0.52674897 0.47520661 0.475      0.63043478 0.41304348 0.37262357
 0.21755725 0.40322581 0.28389831 0.51153846]

Average = 'Macro'
Precision: 0.5002092775053165
Recall: 0.43092772427609327
```

In both the cases the accuracy,precision and recall have reduced on adding the noises. This can be because my self made algorithms of MFCC and spectrogram are not able to train well due to the randomness that have been generated.They are not able to extract the useful features efficiently which is causing a problem in the classification process.