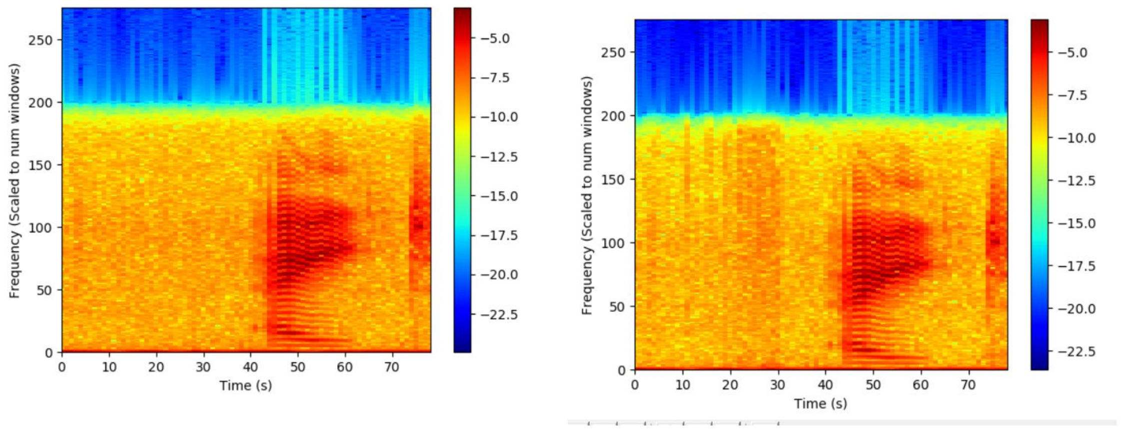
MCA Assignment 2

Deepak Magesh Srivatsav
2016030

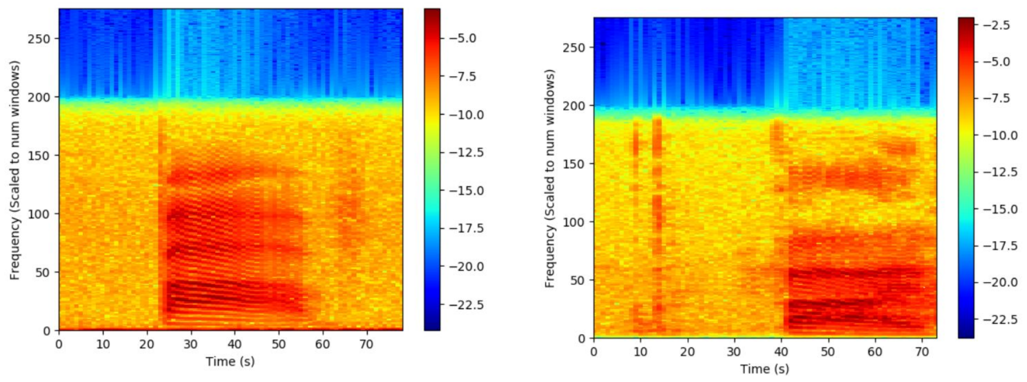
Results are first reported, followed by analysis, and finally instructions to run the code

Part 1 - Spectrogram

Class 8



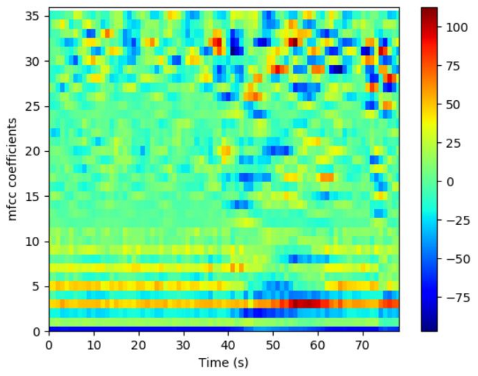
Class 5

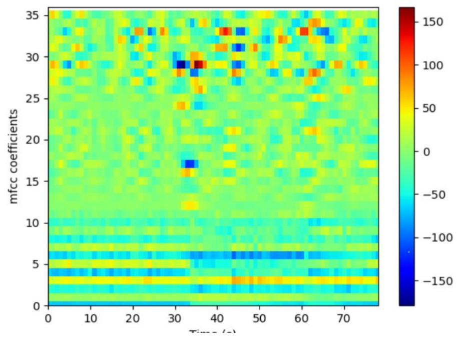


There are clear similarities visible in the spectrograms of samples of class 8 and between those of class 5.

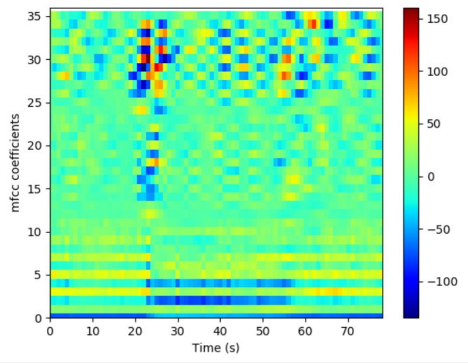
Part 2 - MFCC

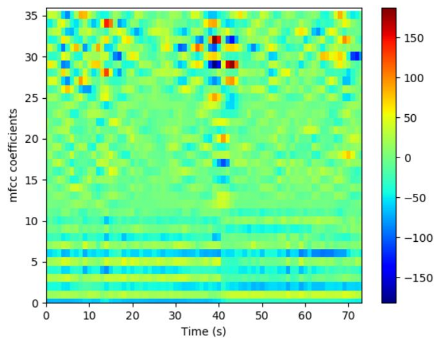
Class 8



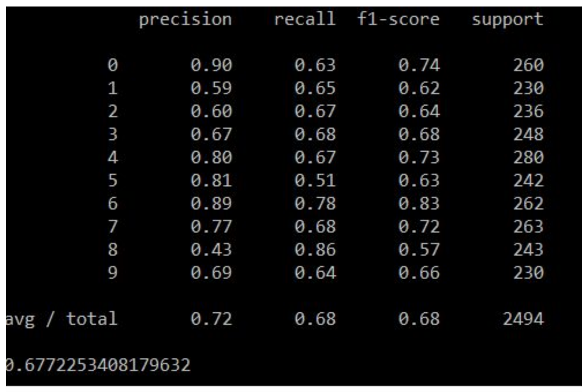


Class 5



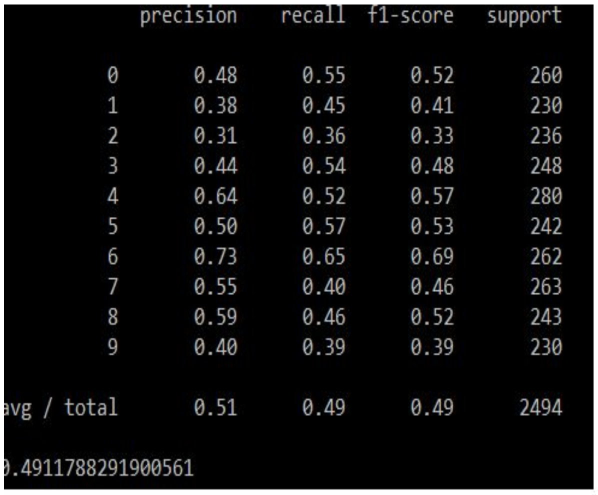


Part 3 - Spectrogram



```
             precision    recall  f1-score   support

          0       0.90      0.63      0.74       260
          1       0.59      0.65      0.62       230
          2       0.60      0.67      0.64       236
          3       0.67      0.68      0.68       248
          4       0.80      0.67      0.73       280
          5       0.81      0.51      0.63       242
          6       0.89      0.78      0.83       262
          7       0.77      0.68      0.72       263
          8       0.43      0.86      0.57       243
          9       0.69      0.64      0.66       230

avg / total       0.72      0.68      0.68      2494

0.6772253408179632
```

SVM parameters - rbf kernel, c=1.
Window size 25ms
Overlap is 50% of window size
Performance without data augmentation on the validation set -
Accuracy - 67.72%
Precision - 0.72
Recall - 0.68
F1 - 0.68

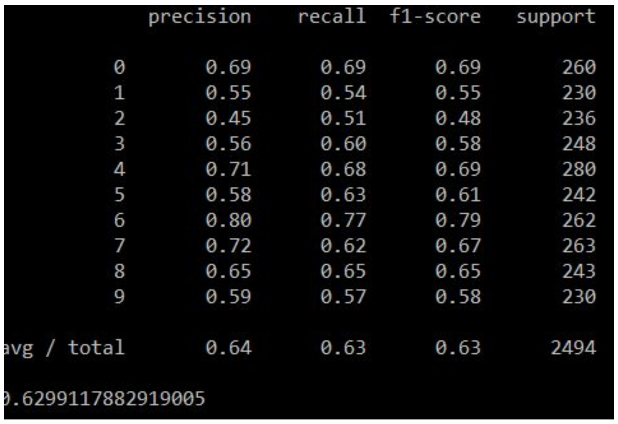


```
             precision    recall  f1-score   support

          0       0.48      0.55      0.52       260
          1       0.38      0.45      0.41       230
          2       0.31      0.36      0.33       236
          3       0.44      0.54      0.48       248
          4       0.64      0.52      0.57       280
          5       0.50      0.57      0.53       242
          6       0.73      0.65      0.69       262
          7       0.55      0.40      0.46       263
          8       0.59      0.46      0.52       243
          9       0.40      0.39      0.39       230

avg / total       0.51      0.49      0.49      2494

.4911788291900561
```

Performance with the following data augmentations -
A linear SVM kernel was used in this case. This is because an rbf kernel with a large amount of data failed to converge after ~12 hours of training.

1) Random noise
2) Pitch modification
3) Speed modification

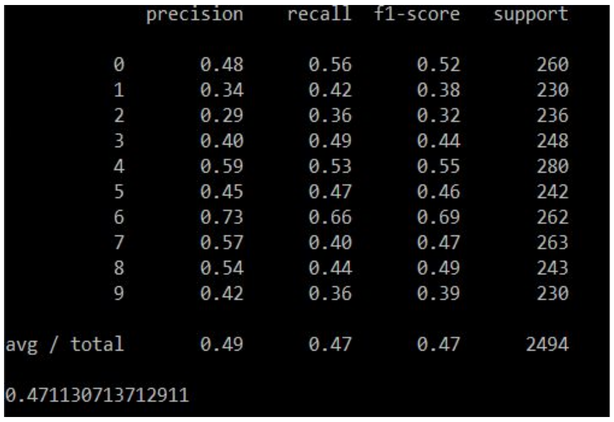
Accuracy - 49.11%
Precision - 0.51
Recall - 0.49
F1 - 0.49



```
             precision    recall  f1-score   support

          0       0.69      0.69      0.69       260
          1       0.55      0.54      0.55       230
          2       0.45      0.51      0.48       236
          3       0.56      0.60      0.58       248
          4       0.71      0.68      0.69       280
          5       0.58      0.63      0.61       242
          6       0.80      0.77      0.79       262
          7       0.72      0.62      0.67       263
          8       0.65      0.65      0.65       243
          9       0.59      0.57      0.58       230

avg / total       0.64      0.63      0.63      2494

.6299117882919005
```

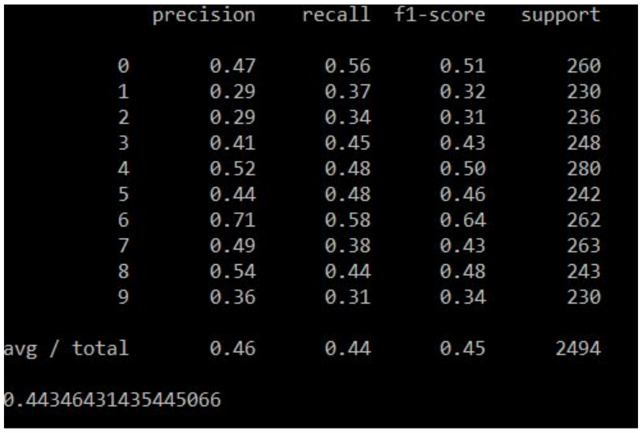
I also tested the spectrogram without augmentations with a linear kernel.
Accuracy - 62.99%
Precision - 0.64
Recall - 0.63
F1 - 0.63

Part 3 - MFCC



```
             precision    recall  f1-score   support

          0       0.48      0.56      0.52       260
          1       0.34      0.42      0.38       230
          2       0.29      0.36      0.32       236
          3       0.40      0.49      0.44       248
          4       0.59      0.53      0.55       280
          5       0.45      0.47      0.46       242
          6       0.73      0.66      0.69       262
          7       0.57      0.40      0.47       263
          8       0.54      0.44      0.49       243
          9       0.42      0.36      0.39       230

avg / total       0.49      0.47      0.47      2494

0.471130713712911
```

SVM parameters - linear kernel, c=0.5
Window size 25ms
Overlap is 50% of window size
Number of ceps = 12
Number of filters = 26
Performance without data augmentation on the validation set -
Accuracy - 47.11%
Precision - 0.49
Recall - 0.47
F1 - 0.47



```
             precision    recall  f1-score   support

          0       0.47      0.56      0.51       260
          1       0.29      0.37      0.32       230
          2       0.29      0.34      0.31       236
          3       0.41      0.45      0.43       248
          4       0.52      0.48      0.50       280
          5       0.44      0.48      0.46       242
          6       0.71      0.58      0.64       262
          7       0.49      0.38      0.43       263
          8       0.54      0.44      0.48       243
          9       0.36      0.31      0.34       230

avg / total       0.46      0.44      0.45      2494

0.44346431435445066
```

Performance with the following data augmentations -
A linear SVM kernel was used in this case.

1) Random noise
2) Pitch modification
3) Speed modification

Accuracy - 44%
Precision - 0.47
Recall - 0.44
F1 - 0.45

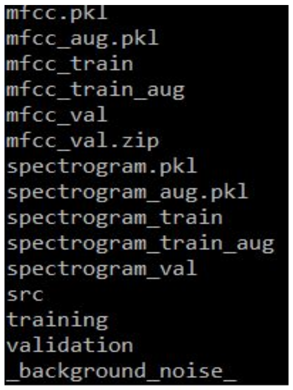
Analysis:

1) Data augmentations did not seem to help much with my experiments. This could perhaps be due to the fact that a linear SVM (not the best classifier!) was used. Using a neural network with LSTMs might have yielded better results on audio data.
2) The performance of the spectrogram was better than that of mfcc. This could be because spectrograms are more feature rich. This however results in a longer training and testing time as well.
3) Adding delta and delta-delta values increased the performance of the mfcc feature based svm.
4) Although kernel SVMs took a lot longer to converge, they performed better for spectrogram features than a linear kernel. However, I observed the opposite for mfcc features. A linear SVM performed better than an rbf kernel based SVM for mfcc features. This might have something to do with the low dimensionality of mfcc data. Furthermore, it is possible that the mfcc features, that are represented as coefficients are to an extent linearly separable.

Instructions:

1) Download features and weights from google drive
2) Unzip features

3) All folders, including models must be placed in the root directory



```
mfcc.pkl
mfcc_aug.pkl
mfcc_train
mfcc_train_aug
mfcc_val
mfcc_val.zip
spectrogram.pkl
spectrogram_aug.pkl
spectrogram_train
spectrogram_train_aug
spectrogram_val
src
training
validation
_background_noise_
```

4) To test the performance of any of the models -
python src/test.py [mfcc/spectrogram] [model_name] [test_dir]
Model name would be the full name, for example, “mfcc.pkl”.
Test dir must have structure similar to train data. The features must be pre generated. If not, features can be generated using question1_1.py or question1_2.py. Note - this may be time consuming depending on the size of the test data.
5) In order to test on unseen data, you must first generate features. For this, you can use the generate_features.py script. This internally calls the functions defined in question1_1 and question1_2.
python src/generate_features.py [mfcc/spectrogram] [data dir] [save dir]
This save dir must then be used along with any of the pretrained models as specified in the 4th point above.
6) To generate spectrogram or mfcc plots -
python src/plot_features.py [spectrogram/mfcc] [feature dir] [num images] [num_classes]
Num images - number of images per class to generate
Num classes - number of classes to generate plots for.
Note: Doing this requires that the pre-generated features are downloaded or generated using the given script