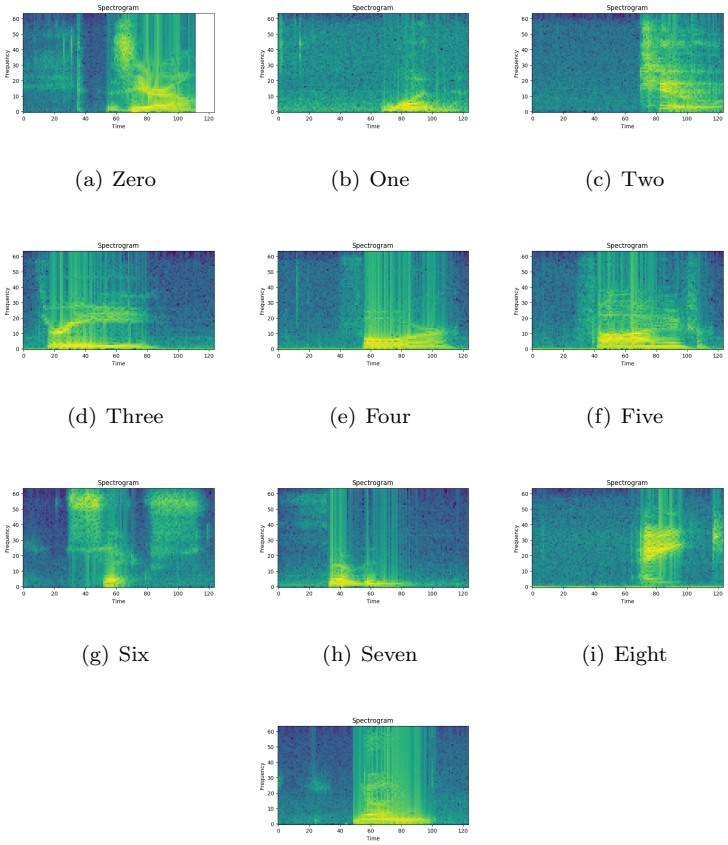
MCA Assignment 2 Report

Vishaal Udandarao
2016119

March 2020

1 Spectrograms

- **Pre-processing**: After reading an audio sample, I perform a padding operation to ensure that every audio sample is 1 second long. That is, at a sampling rate of 16000 Hz, every audio sample is at a consistent length of 16000 samples. The padding is done with a constant zero value throughout. This is done consistently throughout training and validation sets.
- **Algorithm details**: The algorithm implemented takes in a signal (as a time series), a sampling rate, a window size and an overlap (stride) as the parameters and returns the log spectrogram of the signal after processing. I have taken a sampling rate of 16000, a window size of 128 samples (0.008 ms) and an overlap of 0 samples for all training and validation samples. Based on the sampling rate, window size and overlap, I compute the number of output frames by evenly dividing the input signal, and then perform a short time fourier transform (STFT) on the divided output frames. The STFT is performed by computing the fourier coefficients and doubling them, over each individual frame upto the Nyquist sampling limit. Finally, a log (base 10) transform is applied over the coefficient set to smoothen and decorrelate the obtained spectrogram. By using these parameters, the output spectrogram dimensions for a given audio sample is 64x124.
- **Sample plots**: I have plotted one spectrogram from a random audio sample for each of the 10 classes:



(a) Zero (b) One (c) Two

(d) Three (e) Four (f) Five

(g) Six (h) Seven (i) Eight

(j) Nine

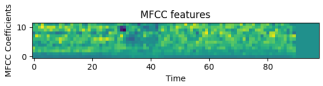
Figure 1: Sample spectrogram feature plots

2 MFCC Features

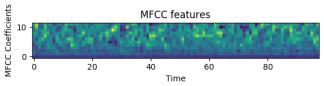
- **Pre-processing**: After reading an audio sample, I perform a padding operation to ensure that every audio sample is 1 second long. That is, at a sampling rate of 16000 Hz, every audio sample is at a consistent length of 16000 samples. The padding is done with a constant zero value throughout. This is done consistently throughout training and validation sets.
- **Algorithm details**: The algorithm implemented takes in a signal (as a time series), a sampling rate, a window size and an overlap (stride) as the parameters and returns the mfcc coefficients of the signal after processing.

I have taken a sampling rate of 16000, a window size of 400 samples (0.025 ms) and an overlap of 160 samples (0.01 ms) for all training and validation samples. Based on the sampling rate, window size and overlap, I compute the number of output frames by evenly dividing the input signal. I then clip the entire set of frames based on the window size and overlap (stride). To have a smooth frame set, I apply the Hamming window function. The Hamming window function applies a non linear mapping from a domain of real inputs to smoothen the input values. Once the hamming transform is applied on the windows, I apply a real FFT (fast-fourier transform) on the computed windows. This gives us the power power spectrum which on converting to mel range gives us the mel frequency spectrum. I then compute the filter banks for each of the windows. Once the filter banks are retrieved, I apply a discrete cosine transform (DCT) to the window frames, which yields quefrency spectrum. I take only the 2nd to 13th quefrencies. This is dont because the other coefficients are not useful for speech recognition due to them representing fast changes in the filter banks. Finally, I apply a liftering option on the obtained quefrencies to filter out unwanted quefrencies. Finally, I apply a mean normalization to the output windows. Thus, finally we obtain the normalized mel frequency cepstral coefficients. For the parameters I've considered, the output MFCC feature size for a given audio sample is 98x12.

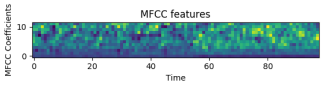
- **Sample plots**: I have plotted one mfcc feature map from a random audio sample for each of the 10 classes:



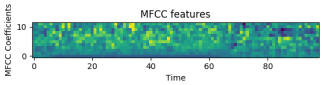
(a) Zero



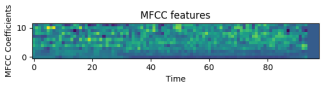
(b) One



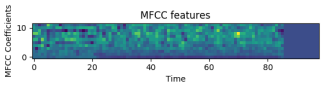
(c) Two



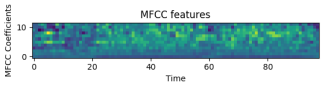
(d) Three



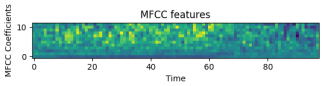
(e) Four



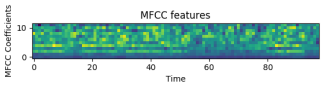
(f) Five



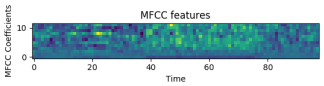
(g) Six



(h) Seven



(i) Eight



(j) Nine

Figure 2: Sample MFCC feature plots

3 SVM training and analysis

3.1 Spectrogram features

The training set contains 10000 samples. On using the spectrogram features, each sample is represented by a feature of 64x124 dimensionality. For training and optimizing the SVM hyperparameters, I perform a gridsearch over the kernel space and slack parameter space. The following hyperparameter values are searched over:

1. C = {0.01, 0.2, 0.5, 1, 2, 5}
2. kernel = {polynomial, linear, rbf, sigmoid}

After performing gridsearch, the optimal hyperparameters were found to be **C=0.5** and **kernel='rbf'**. The top 5 results obtained are shown below:

Parameters	**Accuracy**	**Precision/Recall**
C=0.5, kernel='rbf'	0.642	0.66/0.64
C=0.5, kernel='lin'	0.587	0.59/0.6
C=0.2, kernel='lin'	0.592	0.62/0.6
C=0.2, kernel='rbf'	0.56	0.62/0.56
C=0.2, kernel='poly'	0.54	0.56/0.54

Table 1: Results obtained on validation sets, 'lin' denotes linear kernel, 'poly' denotes polynomial, 'sig' denotes sigmoid kernel, 'rbf' denotes rbf kernel

I have also performed noise augmentation to compare the performance of the trained SVM. For performing noise augmentation, I have used the following steps for each audio file:

- Randomly select one of the 6 six noise audio files
- Randomly choose any index in the range [0, len(noise)-len(signal)]
- Take the indexed noise slice, multiply it with a noise factor (0.005) and add it to the signal
- augmented_signal = signal + 0.005*indexed_noise_signal

On performing noise augmentation, the training set size doubled. Therefore, there were totally 20000 samples, 10000 original samples and 10000 augmented samples. On performing noise augmentation, there was a consistent improvement in the performance of all the grid-search models (2-3% on average).

The below table shows the best model and its corresponding performance improvement on applying noise augmentation (with respect to accuracy, precision, recall):

Model	Accuracy	Precision/Recall
Without noise augmentation	0.642	0.66/0.64
With noise augmentation	0.672	0.68/0.67

The classification report for the best model is:

Classes	Precision	Recall	F1-score
0	0.79	0.67	0.72
1	0.5	0.67	0.57
2	0.55	0.63	0.59
3	0.61	0.65	0.63
4	0.7	0.64	0.66
5	0.77	0.49	0.6
6	0.9	0.86	0.88
7	0.76	0.69	0.72
8	0.69	0.8	0.74
9	0.57	0.62	0.59
micro avg	0.67	0.67	0.67
macro avg	0.68	0.67	0.67
weighted avg	0.69	0.67	0.67

The confusion matrix for the best model is:

```
[[174   9  22   6  15   1   1  24   2   6]
 [  1 154   4   8  19   0   3   0   9  32]
 [ 15  11 149  21  11   0   4   8  12   5]
 [  2  15  28 161   3   3   3   7  15  11]
 [ 13  29  25   4 178  11   0  11   6   3]
 [  3  30   4  18  14 119   3   4  11  36]
 [  0   3   7   3   0   3 226   3  17   0]
 [ 10   8  19   7  13   5   4 181   7   9]
 [  1   6   7  21   1   2   6   0 194   5]
 [  1  44   5  15   2  10   1   1   9 142]]
```

On looking at the class-wise precision values, recall values and confusion matrix, we clearly notice that class 6 performs the best (most number of correct classifications=226). Class 5 has the most number of mis-classifications. Further we see that class 9 is mis-classified as class 2 quite often (most number of pairwise mis-classifications=44). Class 5 is mis-classified as class 9 a lot too (36 mis-classifications)

3.2 MFCC features

The training set contains 10000 samples. On using the MFCC features, each sample is represented by a feature of 98x12 dimensionality. For training and optimizing the SVM hyperparameters, I perform a gridsearch over the kernel space and slack parameter space. The following hyperparameter values are searched over:

1. C = {0.01, 0.2, 0.5, 1, 2, 5}

2. kernel = {polynomial, linear, rbf, sigmoid}

After performing gridsearch, the optimal hyperparameters were found to be **C=0.2** and **kernel='linear'**. The top 5 results obtained are shown below:

Parameters	**Accuracy**	**Precision/Recall**
C=0.2, kernel='lin'	0.562	0.56/0.56
C=1, kernel='lin'	0.523	0.54/0.52
C=0.5, kernel='poly'	0.497	0.49/0.48
C=0.2, kernel='rbf'	0.49	0.57/0.49
C=0.2, kernel='poly'	0.467	0.45/0.46

Table 2: Results obtained on validation sets, 'lin' denotes linear kernel, 'poly' denotes polynomial, 'sig' denotes sigmoid kernel, 'rbf' denotes rbf kernel

I have also performed noise augmentation to compare the performance of the trained SVM. For performing noise augmentation, I have used the same steps as used in the spectrogram features.

On performing noise augmentation, the training set size doubled. Therefore, there were totally 20000 samples, 10000 original samples and 10000 augmented samples. However, after performing noise augmentation, the results seemed to get worse on all the models. There was a drop in the validation set accuracies obtained. The precision and recall values however remained the same for the best model. A possible reason for this might be increase in the variance of the training set distribution, which could not be fully captured by the model.

The below table shows the best model and its corresponding performance degradation on applying noise augmentation (with respect to accuracy, precision, recall):

Model	**Accuracy**	**Precision/Recall**
Without noise augmentation	0.562	0.56/0.56
With noise augmentation	0.558	0.56/0.56

The classification report for the best model is:

Classes	Precision	Recall	F1-score
0	0.67	0.55	0.60
1	0.44	0.51	0.48
2	0.61	0.37	0.46
3	0.40	0.73	0.52
4	0.39	0.72	0.51
5	0.62	0.43	0.51
6	0.89	0.74	0.81
7	0.75	0.58	0.65
8	0.82	0.52	0.64
9	0.58	0.41	0.48
micro avg	0.56	0.56	0.56
macro avg	0.62	0.56	0.57
weighted avg	0.62	0.56	0.57

The confusion matrix for the best model is:

```
[[165   9  27   7  23   5   3  16   2   3]
 [ 13 111  16   4  27  16   5   3   2  33]
 [ 21  21 103  31  10   7   4  11  19   9]
 [ 13  11  43 122   3   4   3  10  30   9]
 [ 29  33  22   3 150  21   2  13   1   6]
 [  7  31   4   4  20 133   7  14   3  19]
 [  4   0   4   3   7   4 211  10  17   2]
 [ 22  12  14   9  16  15  14 153   1   7]
 [  4   4  22  43   5   5  18   2 135   5]
 [  8  36   8  18   5  25   1   9   5 115]]
```

On looking at the class-wise precision values, recall values and confusion matrix, we clearly notice that class 6 performs the best (most number of correct classifications=211). Class 2 has the most number of mis-classifications. Further we see that class 8 is mis-classified as class 3 quite often (most number of pairwise mis-classifications=43). Class 3 is mis-classified as class 2 a lot too (43 mis-classifications). Class 9 is also mis-classified as class 1 a lot (36 mis-classifications).

3.3 Comparison between MFCC and spectrograms

The four best models amongst the entire gird-searched models are ranked as below:

Model	Accuracy	Precision/Recall
Spectrogram (WNA)	0.672	0.68/0.67
Spectrogram (NNA)	0.642	0.66/0.64
MFCC (NNA)	0.562	0.56/0.56
MFCC (WNA)	0.558	0.56/0.56

Table 3: Final ranking of best models of MFCC and spectrogram features. NNA denotes no noise augmentation, WNA denotes with noise augmentation

We clearly see that the models using spectrogram features outperform the models using MFCC features. This might be due to the following reasons:

- In the spectrogram features, I used an overlap parameter of 0 samples, whereas in MFCC features, I used an overlap of 160 samples. This leads to less redundancy in the power spectrum windows and hence lesser correlations in the feature set.
- The window size for spectrogram features was much lesser than the MFCC features. This leads to a much smaller receptive field and more number of windows, which could further lead to more locality of features being preserved in the spectrogram features.

Note: The best models are saved at: https://drive.google.com/open?id=1ss9Z3cfyIanu35hyQTZsvgdmZ2D7HjV