EE679 ASSIGNMENT-3

Agulla Surya Bharath **Roll.No :17D070055**

1. End pointing

I have followed the energy method . Accordingly I have windowed the input audio signal into frames of width 20ms and shifted the frame by 10ms each time. Then measured the energy of each window frame. Thereafter to detect the start frame of the signal I have put a threshold(0.04) on the energy of that frame. So, a first frame with more energy than the threshold(0.04) will be taken as the start frame of the utterance .Then after detecting the start I will reach the end of the audio if the energy in the current frame is less than some threshold(10^{-5} - 10^{-7}) .

Here in some cases I have found that there are multiple peaks and valleys in the energies of audio signal frames , silences in the beginning (and these are not of much at the end of sound signal). So to deal with this I have made following fix:

If the start and end frame number IDs are too close then it means that our end pointers have confused and took the noise between first two silences as the word utterance. So there we go wrong.

So, I have set some threshold(50) on the difference between the start and stop. If the difference between start and stop is below the threshold(50) then ignore the stop point and recompute the next stop point based on the same threshold above.

Sometimes we may not even have any silence at the end. We only have “silence -> noise -> silence -> word utterance -> end “.
To cater this case I have measured the d = e-s and if d< threshold then I am setting the end to the final frame of the audio signal..

The data itself has some noisy samples without any utterances . I have removed them

We face minor problems in endpointing the signal when the word utterance has breaking fashion silence problems while recording the audio.. Window size = 20ms

2. Pre-emphasis of end pointed audio signal.

I have done the preemphasis similar to assignment 2 by passing the end pointed audio signal through the filter H(z) = 1- 0.95*z-1

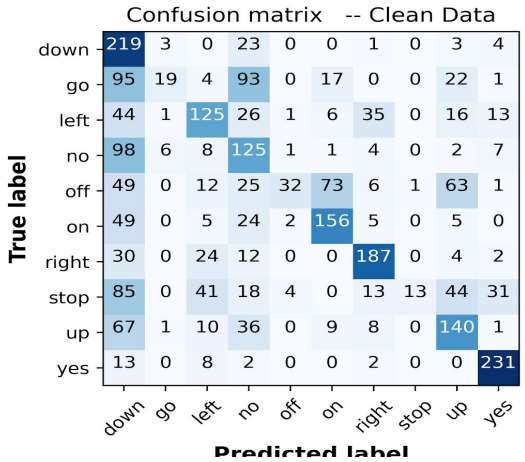
$$Y[n] = X[n] - 0.95 * X[n-1]$$

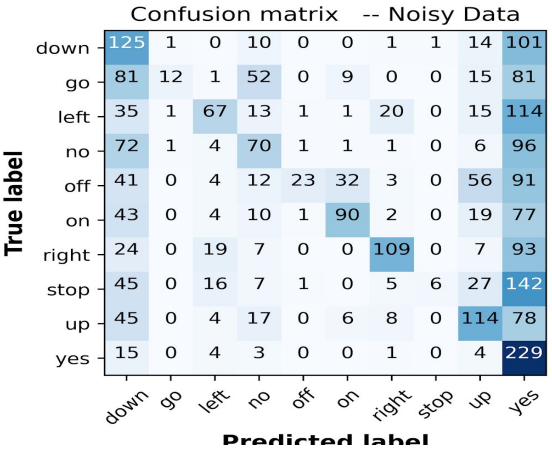
3.I have implemented the statistical method GMM-HMM .(hmmlearn in python)
First I tried to use “`hmm.GMMHMM`” function, the results are as follows :

Tunable hyper parameters in this GMM / HMM model are :

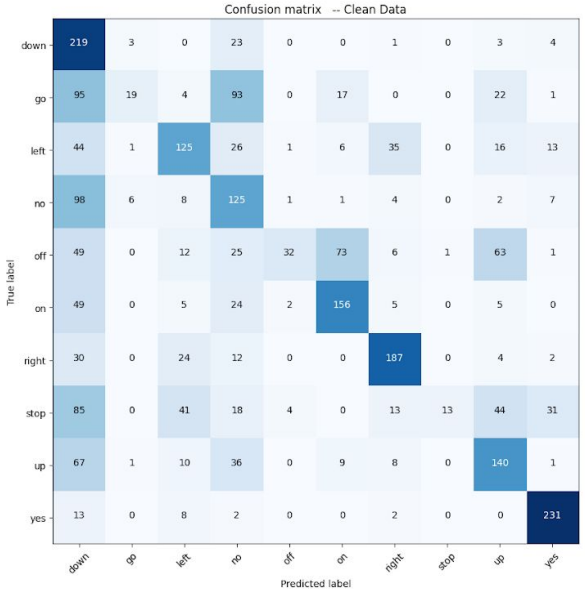
1. Number of iteration of training
2. Number of hidden states in hmm model
3. Number of individual gaussians(in the mixture) in the emissions from each state

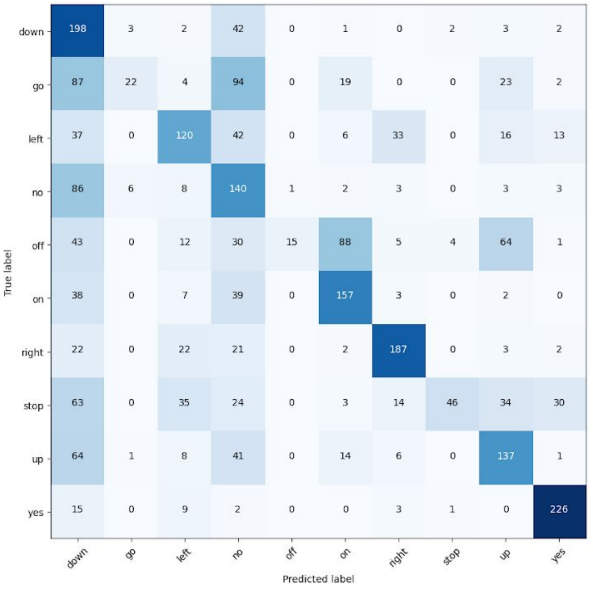
For hmm hidden states = 8 , GMMHMM model.Number of mixes in GMM = 1



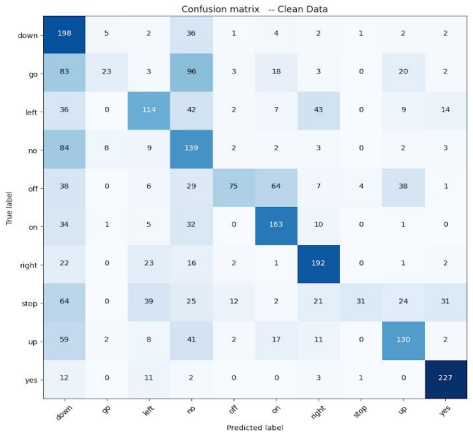


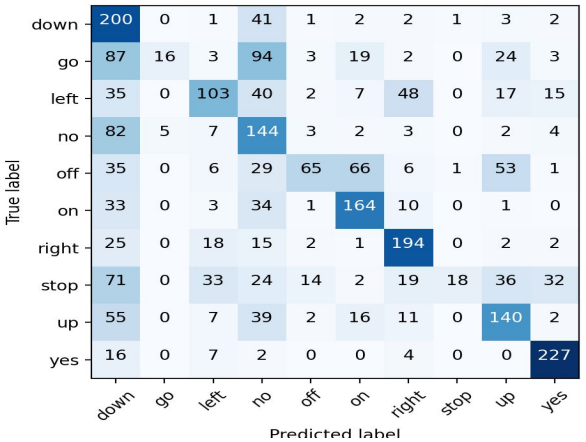
For hmm states = 6 , GMMHMM model. Number of mixes in GMM = 1





For hmm states = 6 , GMMHMM model. Number of mixes in GMM = 2

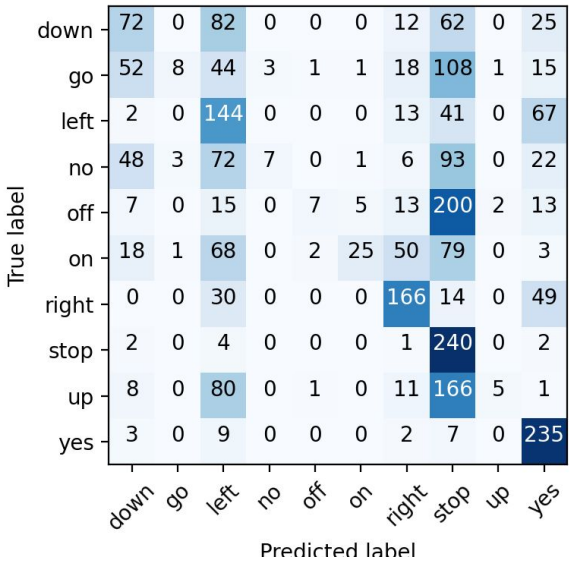


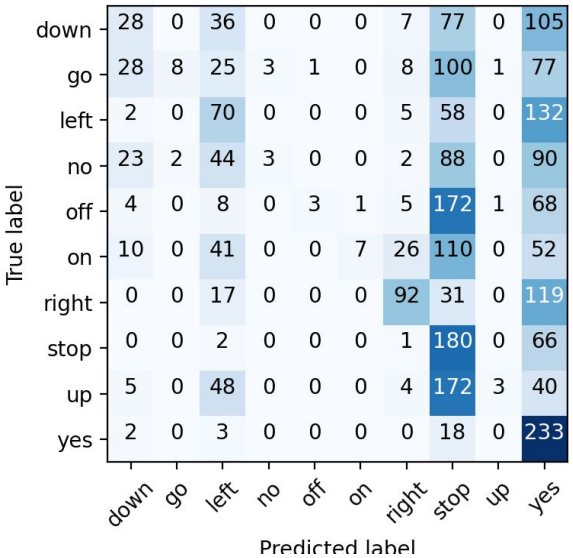


<u>All above cases are for model trained on noiseless / clean data</u> .

Now training on noisy data(I have used “add_noise.py” to add noise) :

For hmm states = 6 ,GMMHMM model. Number of mixes in GMM = 3



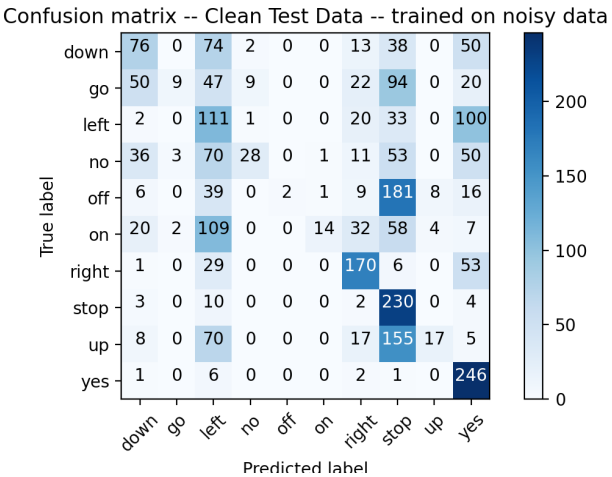


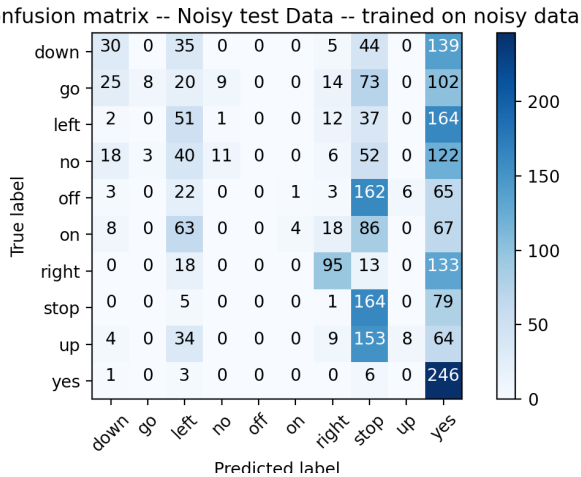
The 10 words are: “down”, “go”, “left”, “no”, “off”, “on”, “right”, “stop”, “up”, “yes”

From the above plots of confusion matrices we can easily see that “go” and “no” are comparatively more confusing.

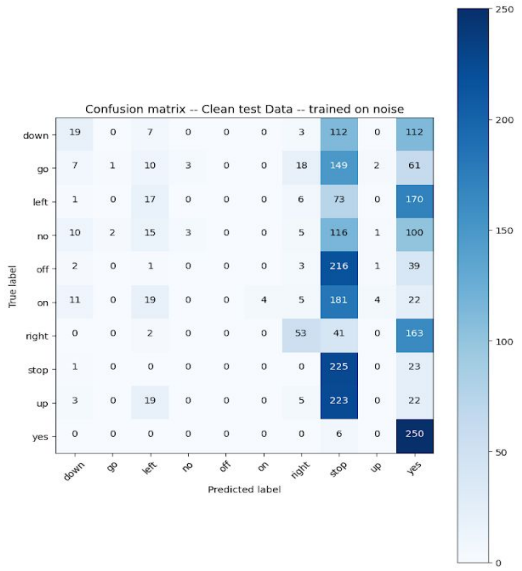
For hmm states = 6 , GMMHMM model. Number of mixes in GMM = 3

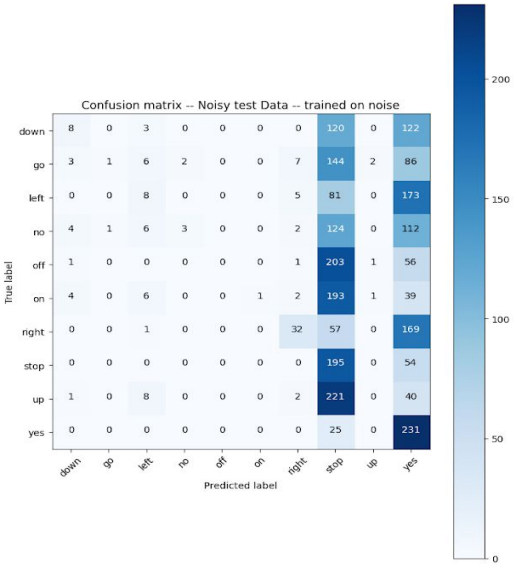
<u>Trained on pink noise addition :</u>





Trained on white noise addition :

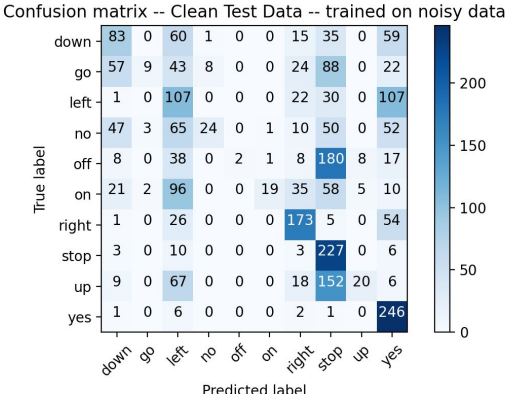


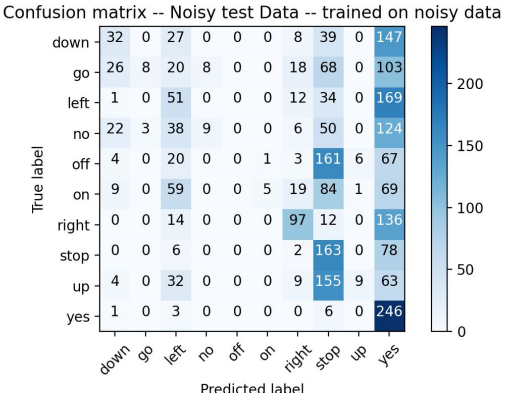


Adding noise requires us to train the model with less gmm mixture for a given number of iterations. We may use more gmm mixture for more iterations (should also be careful to not overfit the model

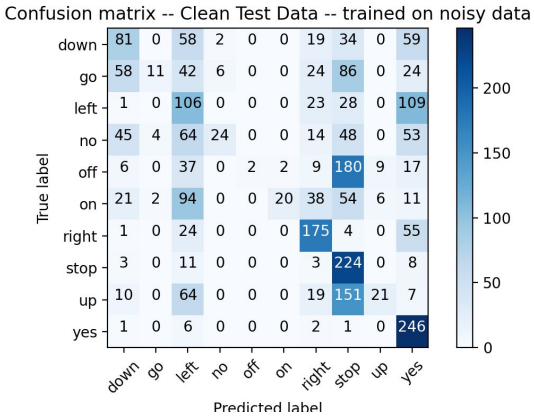
So probably we need more HMM ??

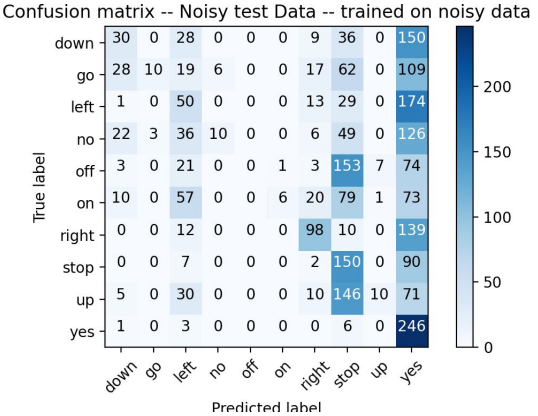
For Number of mixes in GMM = 8





For Number of mixes in GMM = 12

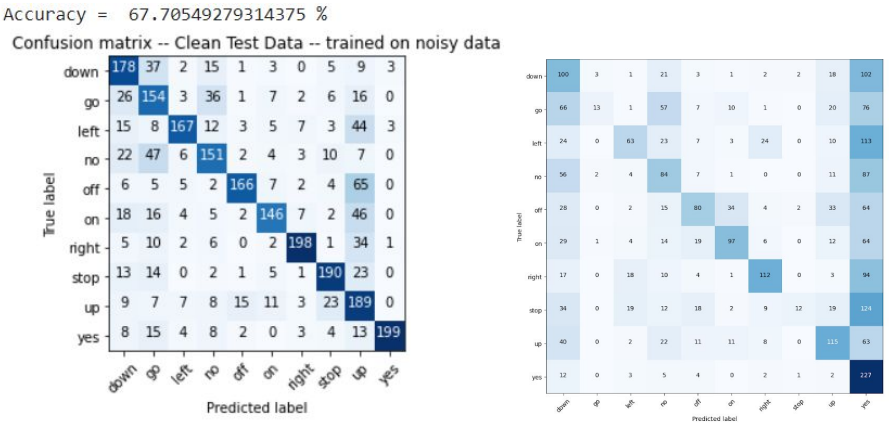




Performance is getting better on clean test data but not on noisy testing data

It looks (also obvious) that training our model on noisy training data is worsening the performance significantly.

I tried to implement multiple configurations of parameters , but due to time constraint and also the code takes 3-4 hrs for running of data. So based on my interpretation above, I have reached the conclusion of an optimal model. For that optimal case the performance is :

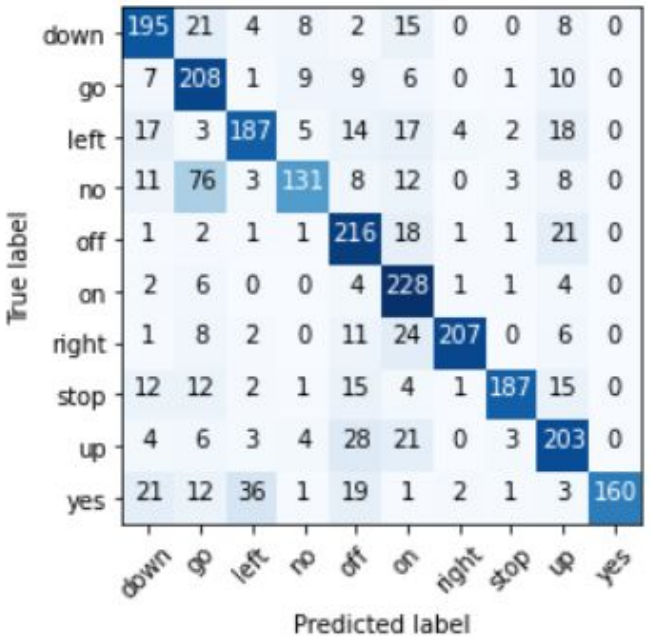


This accuracy is still low even after training for many iterations.
So, I have decided to change the model from h..GMMHMM to hmm.GaussianHMM in hmmlearn.
This GaussianHMM is slow in the training process but significantly increases the performance .

With `hmm.GaussianHMM` used instead of `hmm.GMMHMM`, as model

HMM hidden states = 6 (as concluded above)

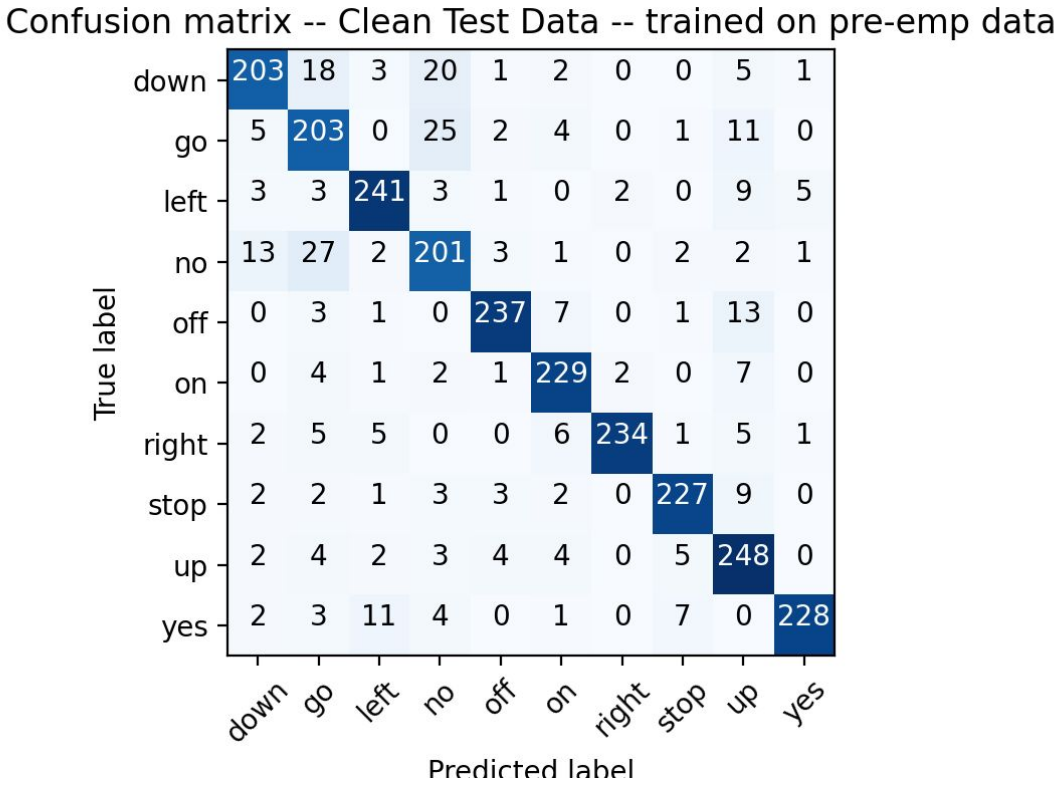
Before pre emphasis **`Accuracy =  74.8733930658356 %`**



Notice the confusion between “go” and “no”

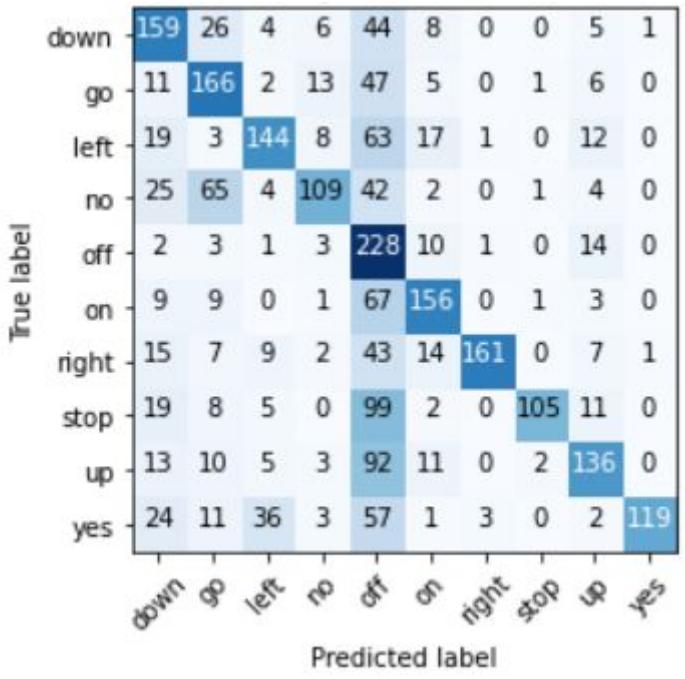
After looking above accuracy , I tried to improve it by pre-processing the test data similar to train data and then doing the prediction based on MFCC feature vectors & trained model.

TASK -A (after pre-emphasis) `Accuracy = 87.68991040124659 %`



"GO" & "NO" are confused more comparatively. This might be because both these both are very small utterances with one vowel sound in common. Also in in both word utterances speakers usually emphasize(more duration) more on the vowel "o" at end, hence cause little confusion.

TASK - B (after pre-emphasis)
Accuracy = 59.32% → Noisy case

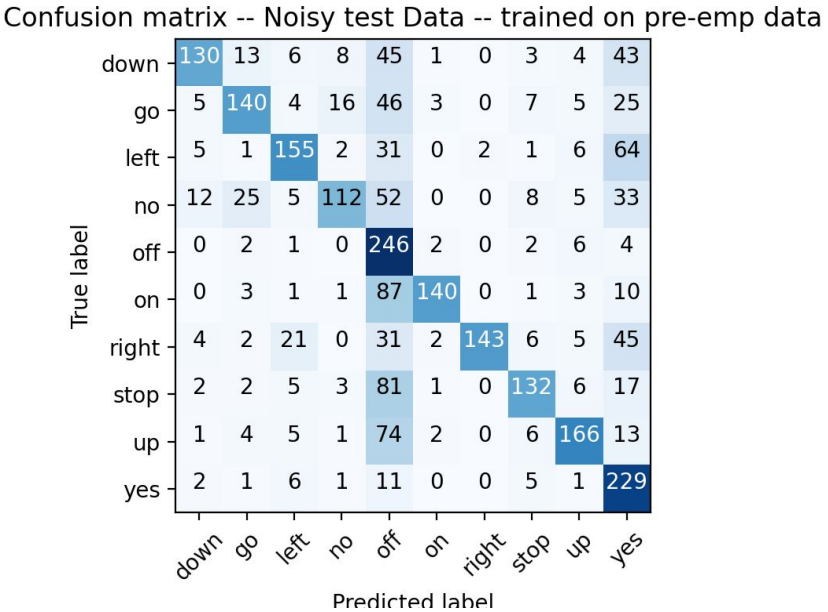


We can see that the “OFF” word is highly confused in cases where we are testing on clean data but it has larger accuracy while testing with noisy data..

Also looks like Gaussian HMM is showing better results than HMMGMM, because if we use GMM then we need to train the model for larger iterations to get better accuracy.

Now,**using noisy training data** to improve accuracy of noisy test cases..

Accuracy = 62.032710280373834 %



OFF is confused in noisy cases..This might be because while uttering OFF we release more air fricated through the mouth for a long duration. We also train the model for OFF with no special care taken against it . This might lead to confusion between the true “OFF” utterance and the noise in the testing data .
Similar arguments goes for the with “YES” too because most of the speakers spend more time on unvoiced sound frication “ss” in the end .This also leads to same confusion as above.
So the utterances with frication and whispering nature tend to be confused more in a noisy environment.

Now using this Gaussian HMM we can try to slightly increase number of HMM hidden states and then train our model for more iterations inorder to achieve even better performance. But needs many iterations of training and hence more time.

NOTE: Due to time and processor constraints I am not able to train my model for more iterations. Increasing the iterations would increase the accuracy significantly but also takes a very long time.
