Audio Classification Assignment

Parampreet Singh
PhD (SPCOM)
Roll No: 21104268
params21@iitk.ac.in

1 Introduction

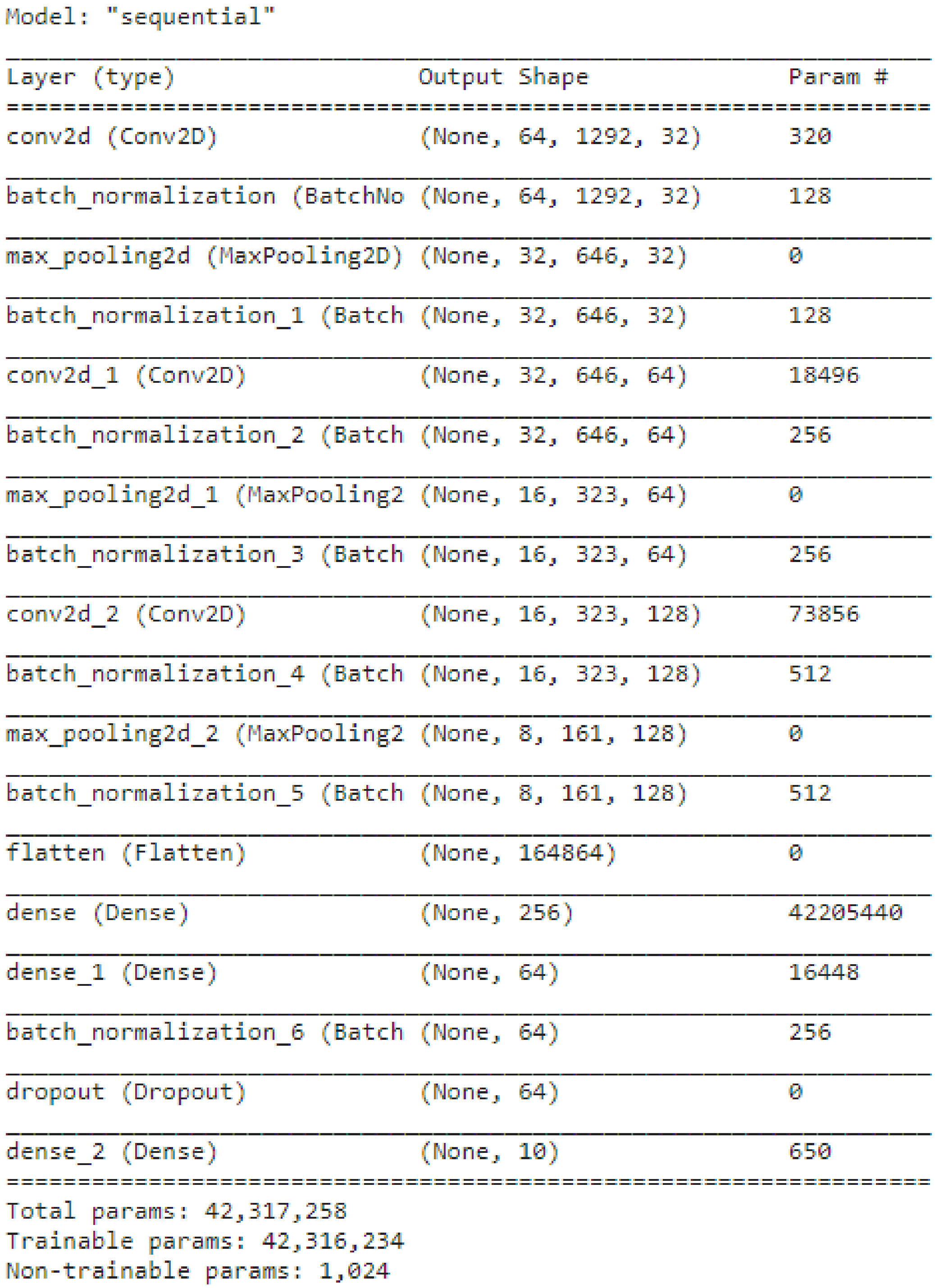
Audio classifications can be of multiple types and forms such as: Acoustic Data Classification or acoustic event detection, Music classification, Natural Language Classification, and Environmental Sound Classification. This domain remains overshadowed by image classification. One of the reasons for that could be the type of pre-processing required for audio is not as straightforward as image data. Usually for audio classification, we can not input the raw audio signal to our deep learning models. So, we need to convert the data to another form which is usually spectrogram, which can preserve temporal and spectral information of the audio signal. Or sometimes, we extract features in the form of mfcc features, in case of music data. To solve a problem of audio classification, we need ample of data as each sound coming from the same source can have temporal variations or spectral differences. To train a reliable machine learning model, we need to train it over every possible data point. But that is not always feasible. Also, audio data being sequential in nature, poses more problems than non-sequential data like images.

2 Literature Survey

Audio machine learning applications used to depend on traditional digital signal processing techniques to extract features. For instance, to understand human speech, audio signals could be analyzed using phonetics concepts to extract elements like phonemes. All of this required a lot of domain-specific expertise to solve these problems and tune the system for better performance. For image data, it is desirable to use CNN. The same has been proven repeatedly since the advent of CNNs. As spectrograms are also kind if images only, we can use CNN on those. But as we are talking about sequential data, CNNs followed by RNNs or LSTMs could perform better as they would preserve the temporal information also. But as RNNs or LSTMs are not covered in the class yet, we are not allowed to use them. In our case, we can also try getting out mfcc features out of the spectrograms given using librosa library and check how well the model performs. Also, as the number of samples is just 100 per class, the possibility of getting a very good model is very less. However, data augmentation may help in this case.

3 Methods Used:

The given dataset was 1000 samples which were already in the form of spectrograms created by using Nfft = 2048, nmels = 128, hop length = 512, windowing function = hann. The dataset required a lot of pre-processing. First thing I had to do is to do zero padding as all the audios were not of equal length. So, zero padding was done to all the samples of spectrograms, making their shape as(128,2584). Also, some pre-processing had to be done to convert the dataset to a shape of (1000,128,2584,1), where 1000 is the number of samples, 128 is the number of frequency values corresponding to each frame of spectrogram, 2584 is the number of frames in each sample and 1 represents greyscale image. After zero-padding, we can use this dataset in our first model.



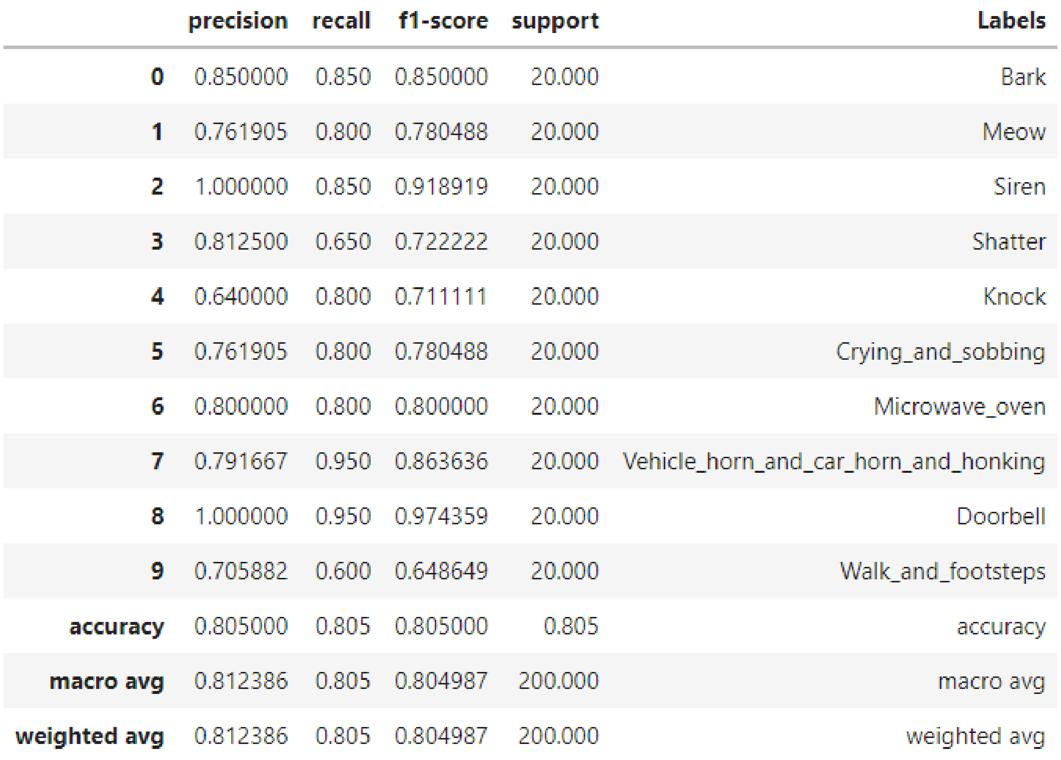
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d (Conv2D)              (None, 64, 1292, 32)      320       
_________________________________________________________________
batch_normalization (BatchNo (None, 64, 1292, 32)      128       
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 32, 646, 32)       0         
_________________________________________________________________
batch_normalization_1 (Batch (None, 32, 646, 32)       128       
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 32, 646, 64)       18496     
_________________________________________________________________
batch_normalization_2 (Batch (None, 32, 646, 64)       256       
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 16, 323, 64)       0         
_________________________________________________________________
batch_normalization_3 (Batch (None, 16, 323, 64)       256       
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 16, 323, 128)      73856     
_________________________________________________________________
batch_normalization_4 (Batch (None, 16, 323, 128)      512       
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 8, 161, 128)       0         
_________________________________________________________________
batch_normalization_5 (Batch (None, 8, 161, 128)       512       
_________________________________________________________________
flatten (Flatten)            (None, 164864)            0         
_________________________________________________________________
dense (Dense)                (None, 256)               42205440  
_________________________________________________________________
dense_1 (Dense)              (None, 64)                16448     
_________________________________________________________________
batch_normalization_6 (Batch (None, 64)                256       
_________________________________________________________________
dropout (Dropout)            (None, 64)                0         
_________________________________________________________________
dense_2 (Dense)              (None, 10)                650       
=================================================================
Total params: 42,317,258
Trainable params: 42,316,234
Non-trainable params: 1,024
```

Figure 1: Model Architecture for model 1

3.1 Method 1:

As my first model, I built a sequential CNN model with specifications as shown in the figure 1.

This model is supposed to work very well as CNN is good for images and the spectrograms here can be considered as images only. We feed the whole datasets with a batch size of 32 and trained for 40 epochs. The training loss decreased very quickly, validation loss also decreased gradually and a good overall accuracy of 0.80 was achieved with precision and recall close to 1 for some lables as shown in the figure 2.



	precision	recall	f1-score	support	Labels
0	0.850000	0.850	0.850000	20.000	Bark
1	0.761905	0.800	0.780488	20.000	Meow
2	1.000000	0.850	0.918919	20.000	Siren
3	0.812500	0.650	0.722222	20.000	Shatter
4	0.640000	0.800	0.711111	20.000	Knock
5	0.761905	0.800	0.780488	20.000	Crying_and_sobbing
6	0.800000	0.800	0.800000	20.000	Microwave_oven
7	0.791667	0.950	0.863636	20.000	Vehicle_horn_and_car_horn_and_honking
8	1.000000	0.950	0.974359	20.000	Doorbell
9	0.705882	0.600	0.648649	20.000	Walk_and_footsteps
accuracy	0.805000	0.805	0.805000	0.805	accuracy
macro avg	0.812386	0.805	0.804987	200.000	macro avg
weighted avg	0.812386	0.805	0.804987	200.000	weighted avg

Figure 2: Various Scores on Validation Set for model 1

Now, the model was tested on an unseen test data. As we can see, an accuracy of around 0.68 was observed as in figure 3. Precision and recall are also good for some cases but very bad for others. The f1score for walk_and_footsteps and horn_honking are the lowest. It could be because there are supposed to be sequential in nature. CNN might not have been able to learn their sequential nature.

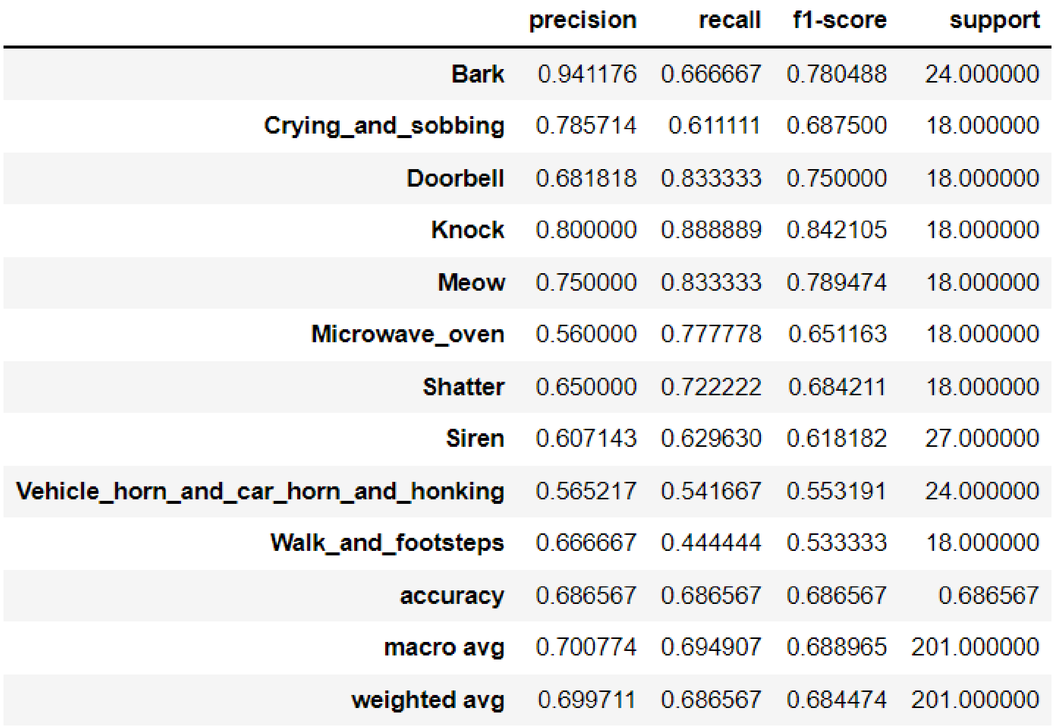
3.2 Method 2:

In this model, I used a simple dense neural network, with specifications as shown in the figure 4. This model unexpectedly trained really good. Trained it for 50 epochs and each time both validation loss and training loss reduced until finally it almost settled down.

Its validation accuracy as seen in the figure 5 is reported to be good with walk_and_footsteps again with the lowest f1score. If we see the test accuracy as in figure 6, it was about 0.57 only with walk_and_footsteps again with the lowest f1score because of the same reason.

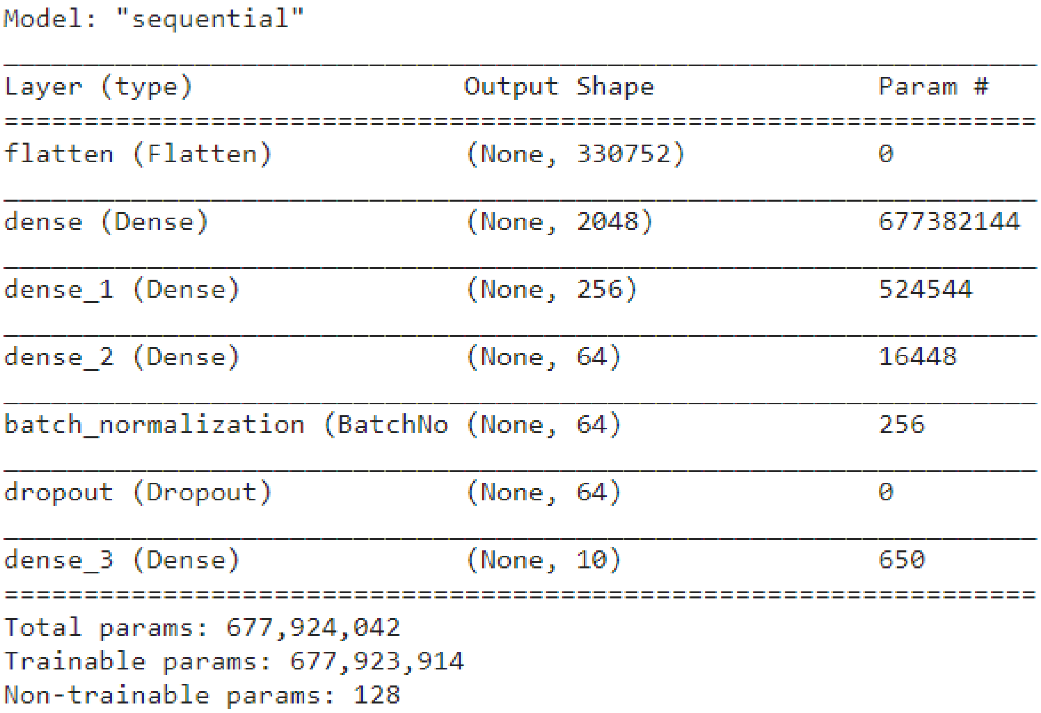
3.3 Method 3:

This time, I extracted mfcc features with n_mfcc = 16 from the mel spectrogram given, using a pre-defined function in librosa. Although mfcc features are supposed to work better for music data



	precision	recall	f1-score	support
Bark	0.941176	0.666667	0.780488	24.000000
Crying_and_sobbing	0.785714	0.611111	0.687500	18.000000
Doorbell	0.681818	0.833333	0.750000	18.000000
Knock	0.800000	0.888889	0.842105	18.000000
Meow	0.750000	0.833333	0.789474	18.000000
Microwave_oven	0.560000	0.777778	0.651163	18.000000
Shatter	0.650000	0.722222	0.684211	18.000000
Siren	0.607143	0.629630	0.618182	27.000000
Vehicle_horn_and_car_horn_and_honking	0.565217	0.541667	0.553191	24.000000
Walk_and_footsteps	0.666667	0.444444	0.533333	18.000000
accuracy	0.686567	0.686567	0.686567	0.686567
macro avg	0.700774	0.694907	0.688965	201.000000
weighted avg	0.699711	0.686567	0.684474	201.000000

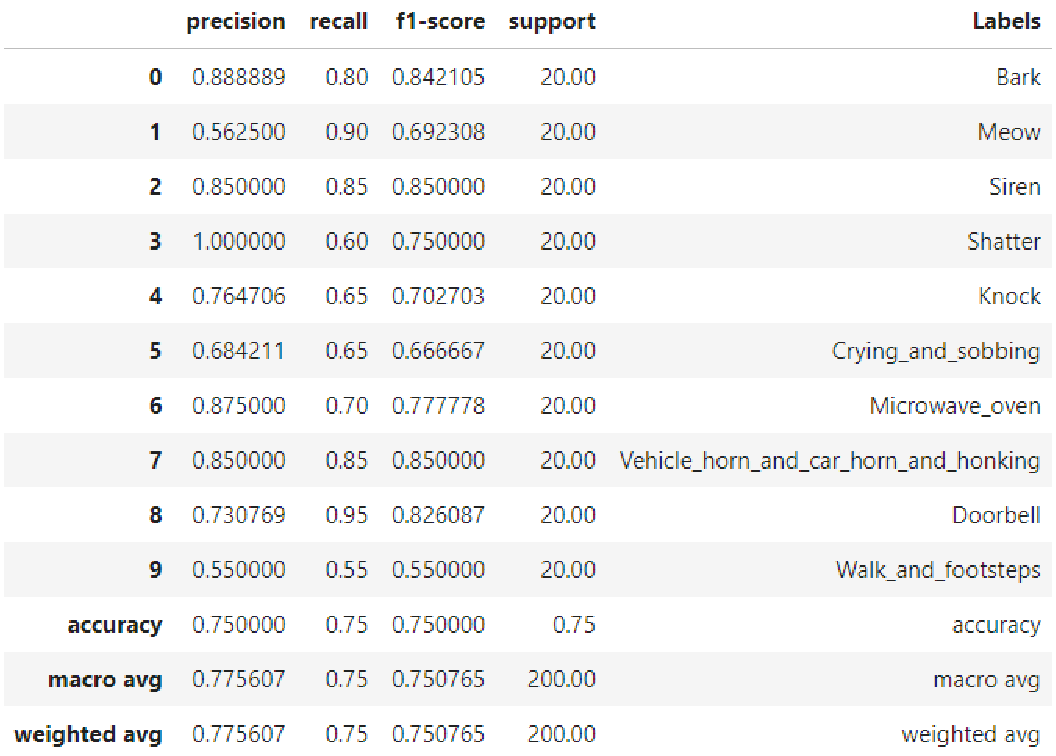
Figure 3: Various Scores on Test Set for model 1



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
flatten (Flatten)            (None, 330752)            0         
_________________________________________________________________
dense (Dense)                (None, 2048)              677382144 
_________________________________________________________________
dense_1 (Dense)              (None, 256)               524544    
_________________________________________________________________
dense_2 (Dense)              (None, 64)                16448     
_________________________________________________________________
batch_normalization (BatchNo (None, 64)                256       
_________________________________________________________________
dropout (Dropout)            (None, 64)                0         
_________________________________________________________________
dense_3 (Dense)              (None, 10)                650       
=================================================================
Total params: 677,924,042
Trainable params: 677,923,914
Non-trainable params: 128
```

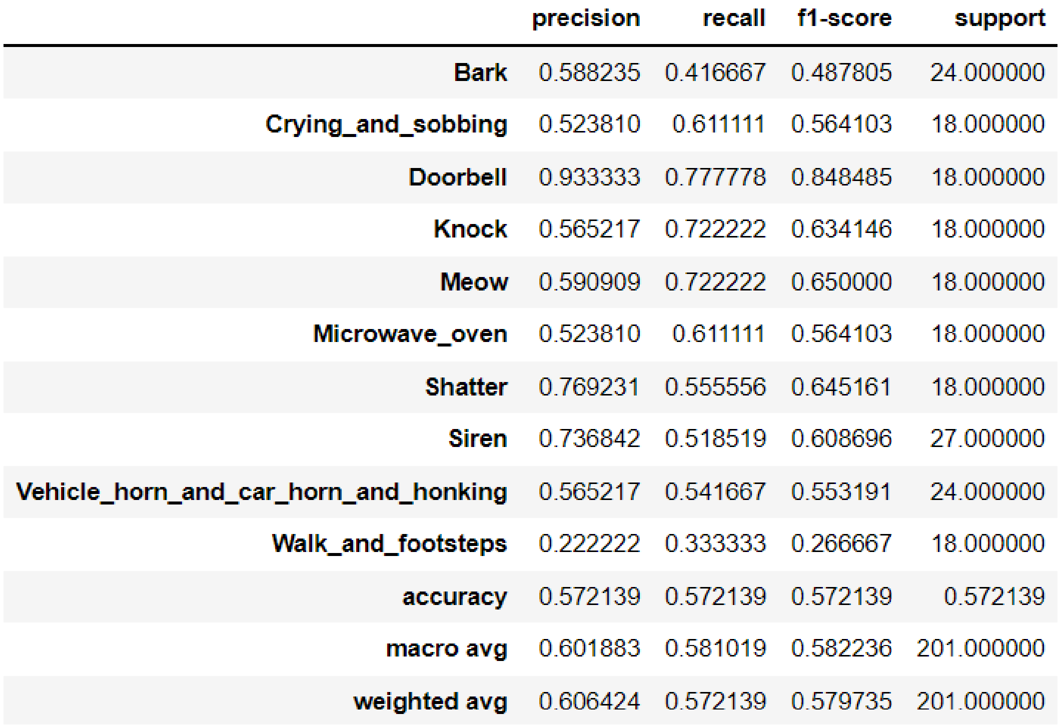
Figure 4: Model Architecture for model 2

as these audio could be well beyond the hearing range also and mfcc is known tocapture very high frequency sounds badly. We will see how it works if we apply a CNN network after extracting mfcc



	precision	recall	f1-score	support	Labels
0	0.888889	0.80	0.842105	20.00	Bark
1	0.562500	0.90	0.692308	20.00	Meow
2	0.850000	0.85	0.850000	20.00	Siren
3	1.000000	0.60	0.750000	20.00	Shatter
4	0.764706	0.65	0.702703	20.00	Knock
5	0.684211	0.65	0.666667	20.00	Crying_and_sobbing
6	0.875000	0.70	0.777778	20.00	Microwave_oven
7	0.850000	0.85	0.850000	20.00	Vehicle_horn_and_car_horn_and_honking
8	0.730769	0.95	0.826087	20.00	Doorbell
9	0.550000	0.55	0.550000	20.00	Walk_and_footsteps
accuracy	0.750000	0.75	0.750000	0.75	accuracy
macro avg	0.775607	0.75	0.750765	200.00	macro avg
weighted avg	0.775607	0.75	0.750765	200.00	weighted avg

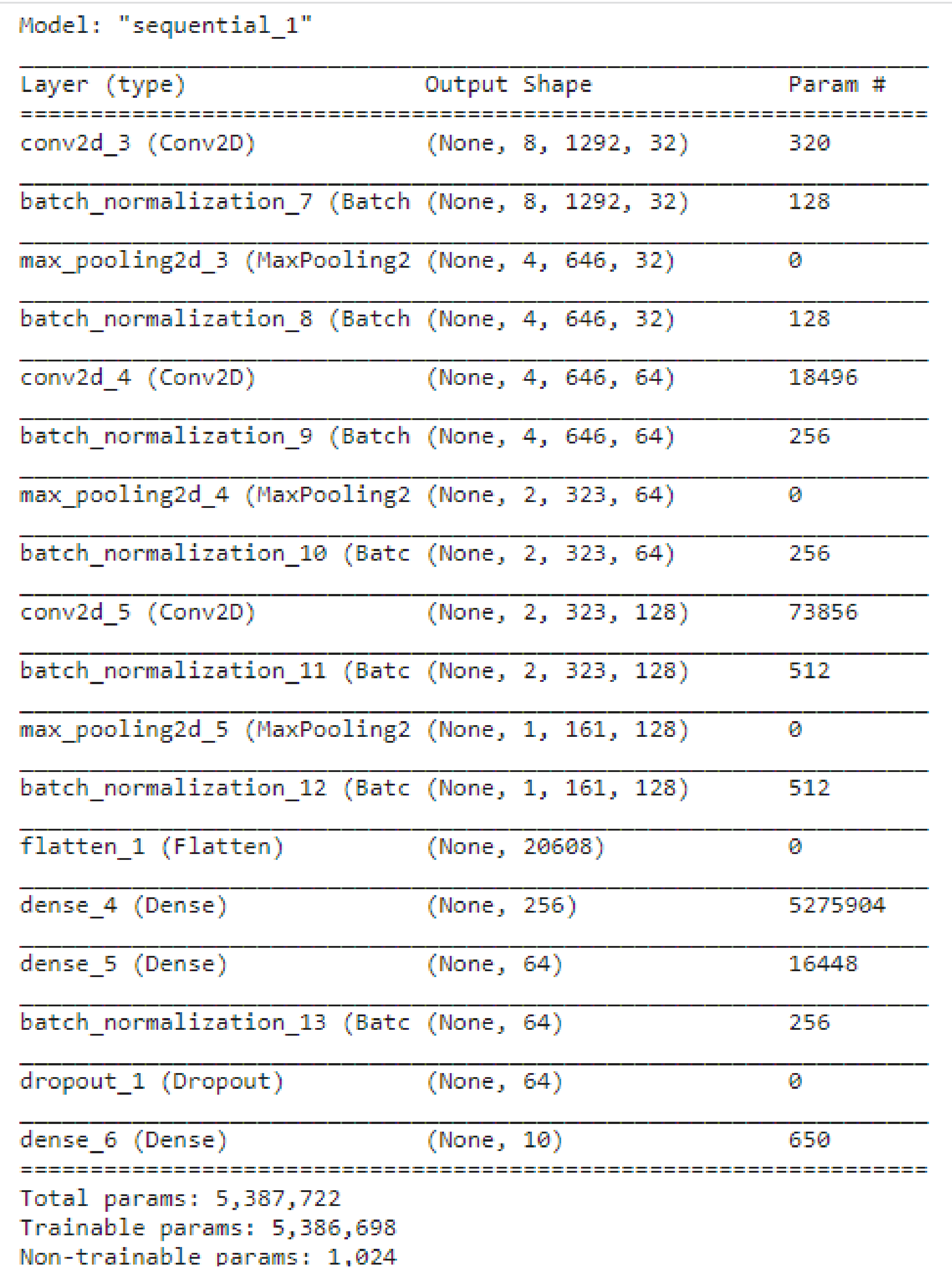
Figure 5: Various Scores on Validation Set for model 2



	precision	recall	f1-score	support
Bark	0.588235	0.416667	0.487805	24.000000
Crying_and_sobbing	0.523810	0.611111	0.564103	18.000000
Doorbell	0.933333	0.777778	0.848485	18.000000
Knock	0.565217	0.722222	0.634146	18.000000
Meow	0.590909	0.722222	0.650000	18.000000
Microwave_oven	0.523810	0.611111	0.564103	18.000000
Shatter	0.769231	0.555556	0.645161	18.000000
Siren	0.736842	0.518519	0.608696	27.000000
Vehicle_horn_and_car_horn_and_honking	0.565217	0.541667	0.553191	24.000000
Walk_and_footsteps	0.222222	0.333333	0.266667	18.000000
accuracy	0.572139	0.572139	0.572139	0.572139
macro avg	0.601883	0.581019	0.582236	201.000000
weighted avg	0.606424	0.572139	0.579735	201.000000

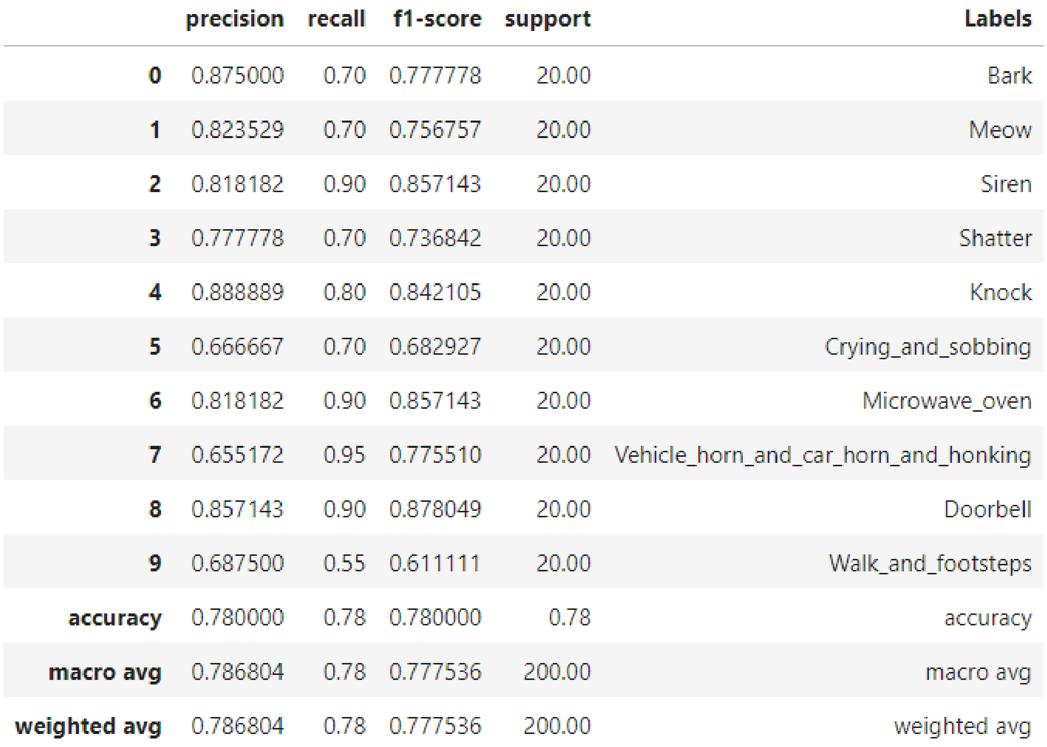
Figure 6: Various Scores on Test Set for model 2

features from the input dataset. A model as shown in the figure 7 was made and trained using mfcc features. This method got a good training and validation accuracy after training for about 50 epochs.



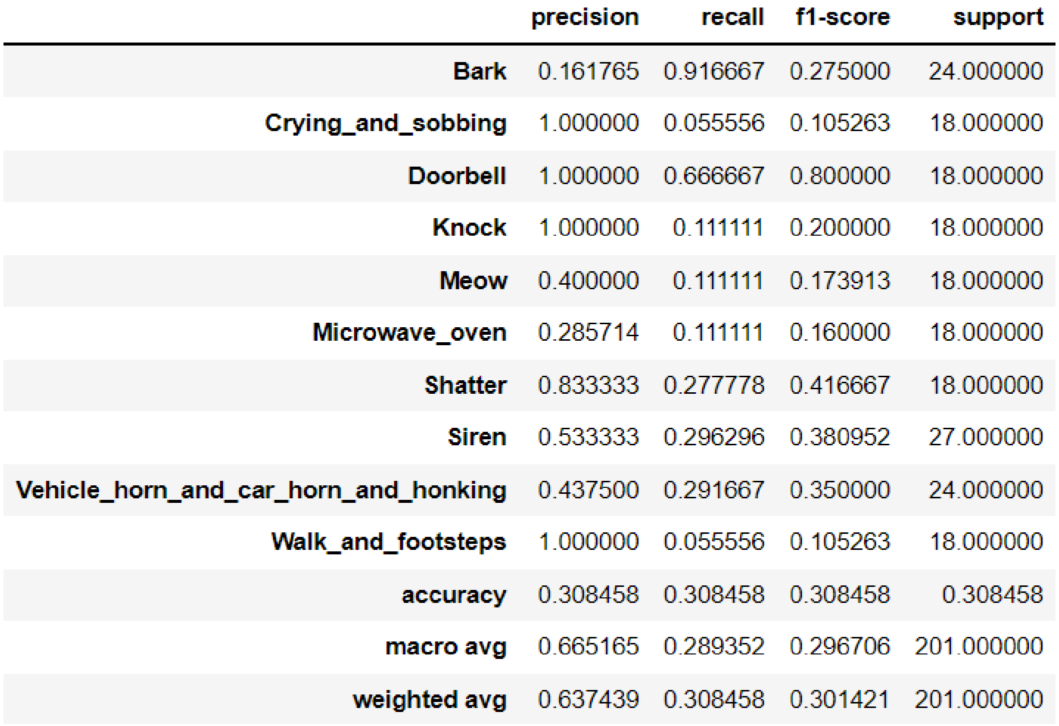
```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_3 (Conv2D)            (None, 8, 1292, 32)       320       
_________________________________________________________________
batch_normalization_7 (Batch (None, 8, 1292, 32)       128       
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 4, 646, 32)        0         
_________________________________________________________________
batch_normalization_8 (Batch (None, 4, 646, 32)        128       
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 4, 646, 64)        18496     
_________________________________________________________________
batch_normalization_9 (Batch (None, 4, 646, 64)        256       
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 2, 323, 64)        0         
_________________________________________________________________
batch_normalization_10 (Batc (None, 2, 323, 64)        256       
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 2, 323, 128)       73856     
_________________________________________________________________
batch_normalization_11 (Batc (None, 2, 323, 128)       512       
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 1, 161, 128)       0         
_________________________________________________________________
batch_normalization_12 (Batc (None, 1, 161, 128)       512       
_________________________________________________________________
flatten_1 (Flatten)          (None, 20608)             0         
_________________________________________________________________
dense_4 (Dense)              (None, 256)               5275904   
_________________________________________________________________
dense_5 (Dense)              (None, 64)                16448     
_________________________________________________________________
batch_normalization_13 (Batc (None, 64)                256       
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0         
_________________________________________________________________
dense_6 (Dense)              (None, 10)                650       
=================================================================
Total params: 5,387,722
Trainable params: 5,386,698
Non-trainable params: 1,024
```

Figure 7: Model Architecture for model 3



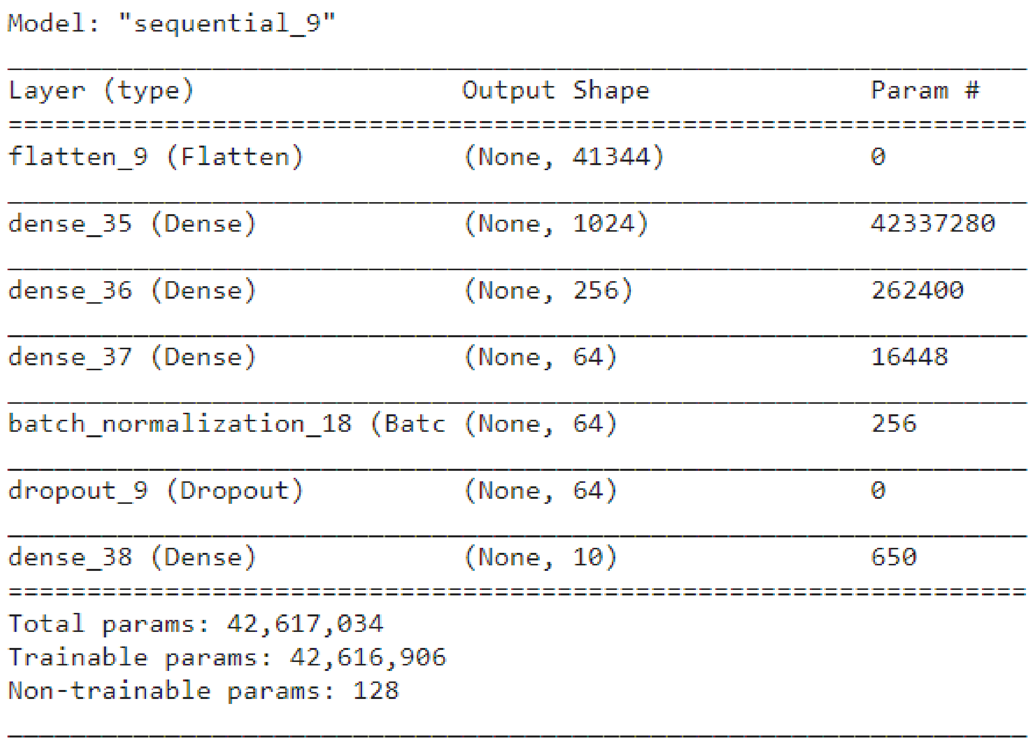
	precision	recall	f1-score	support	Labels
0	0.875000	0.70	0.777778	20.00	Bark
1	0.823529	0.70	0.756757	20.00	Meow
2	0.818182	0.90	0.857143	20.00	Siren
3	0.777778	0.70	0.736842	20.00	Shatter
4	0.888889	0.80	0.842105	20.00	Knock
5	0.666667	0.70	0.682927	20.00	Crying_and_sobbing
6	0.818182	0.90	0.857143	20.00	Microwave_oven
7	0.655172	0.95	0.775510	20.00	Vehicle_horn_and_car_horn_and_honking
8	0.857143	0.90	0.878049	20.00	Doorbell
9	0.687500	0.55	0.611111	20.00	Walk_and_footsteps
accuracy	0.780000	0.78	0.780000	0.78	accuracy
macro avg	0.786804	0.78	0.777536	200.00	macro avg
weighted avg	0.786804	0.78	0.777536	200.00	weighted avg

Figure 8: Various Scores on Validation Set for model 3



	precision	recall	f1-score	support
Bark	0.161765	0.916667	0.275000	24.000000
Crying_and_sobbing	1.000000	0.055556	0.105263	18.000000
Doorbell	1.000000	0.666667	0.800000	18.000000
Knock	1.000000	0.111111	0.200000	18.000000
Meow	0.400000	0.111111	0.173913	18.000000
Microwave_oven	0.285714	0.111111	0.160000	18.000000
Shatter	0.833333	0.277778	0.416667	18.000000
Siren	0.533333	0.296296	0.380952	27.000000
Vehicle_horn_and_car_horn_and_honking	0.437500	0.291667	0.350000	24.000000
Walk_and_footsteps	1.000000	0.055556	0.105263	18.000000
accuracy	0.308458	0.308458	0.308458	0.308458
macro avg	0.665165	0.289352	0.296706	201.000000
weighted avg	0.637439	0.308458	0.301421	201.000000

Figure 9: Various Scores on Test Set for model 3



```
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
flatten_9 (Flatten)          (None, 41344)             0         
_________________________________________________________________
dense_35 (Dense)             (None, 1024)              42337280  
_________________________________________________________________
dense_36 (Dense)             (None, 256)               262400    
_________________________________________________________________
dense_37 (Dense)             (None, 64)                16448     
_________________________________________________________________
batch_normalization_18 (Batc (None, 64)                256       
_________________________________________________________________
dropout_9 (Dropout)          (None, 64)                0         
_________________________________________________________________
dense_38 (Dense)             (None, 10)                650       
=================================================================
Total params: 42,617,034
Trainable params: 42,616,906
Non-trainable params: 128
_________________________________________________________________
```

Figure 10: Model Architecture for model 4

Validation Accuracy is reported to be 0.78 which is a decent value in figure 8. As for the f1 score, again the same walk_and_footsteps label had the lowest value. It looks like there were less than enough samples for this label and also the samples could be mostly variable or some features may be resembling to some other class that is why predictions are coming out to be the worst for this one.

If we have a look at the test accuracy as in figure 9, it was only around 0.3. Such a low value was a bit of a shock. It performed really well for crying_and_sobbing and Dorbell. Crying_and_sobbing being closer to human frequency range could be the reason, but it is still a high frequency audio.

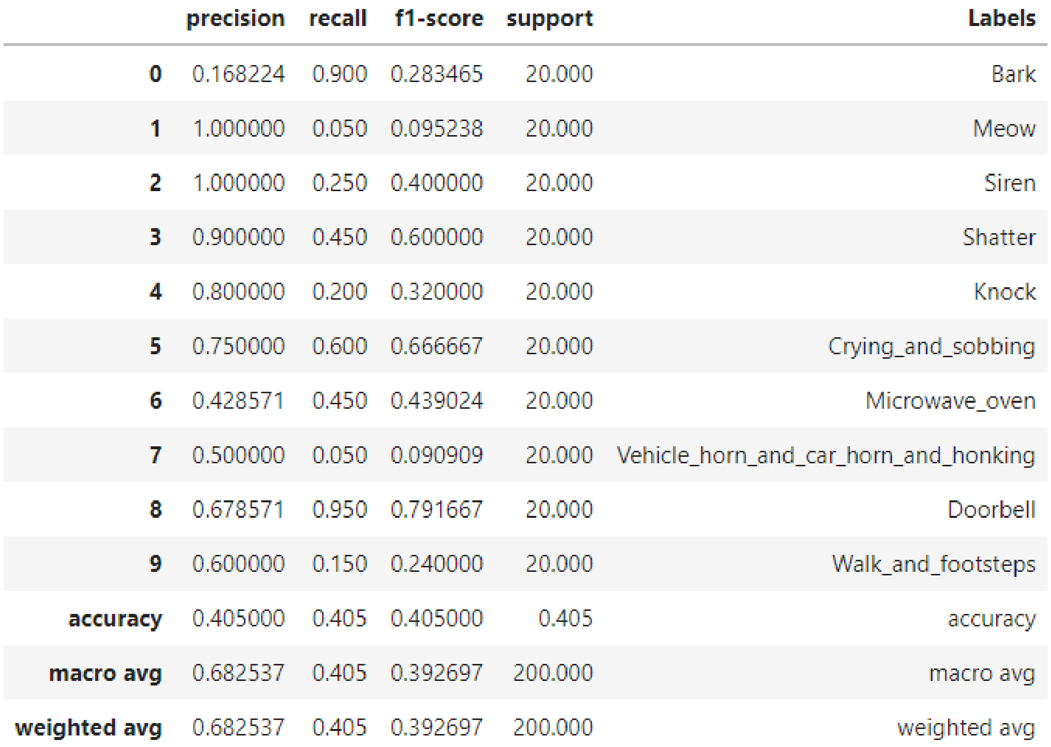
3.4 Method 4:

This time we had a simple ANN on mfcc Features extracted. The model is shown in figure 10. This model just did not seem to work well at all even after changing different hyperparameters. After training for about 50 epochs, validation loss would not reduce much in each step and validation accuracy was reported to be around 0.4 only in figure 11.

In this case, it was obvious for the model to perform bad on the test set. It had an accuracy of 0.2 with siren and walk_and_footsteps not being predicted even once as seen in figure 12.

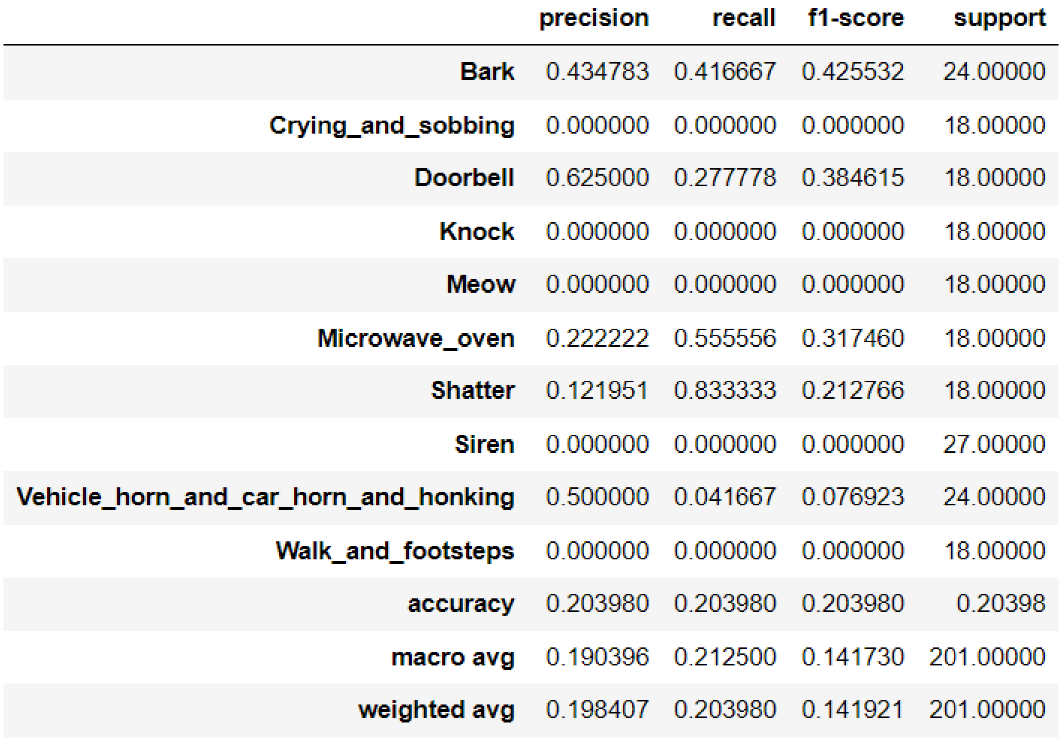
3.5 Method 5:

Now this time around, we will try data augmentation which seems to be the way out here. Although a lot of other options are there for data augmentation, but I went with just temporal shift. Ideally for best results, I should have made a number of copies of the same sample by adding some temporal shift or some minor noise and create around 10 times more data samples for best results. But I just padded zeros in the front instead of the back and created 2000 total samples. Now, er have 200 samples for each. As the code was getting heavy with augmentation, I only created 100 extra samples per sample, otherwise the runtime would crash each time as it would run out of ram. Then a model was created as shown in figure 13. This model performed really well as compared to all the models



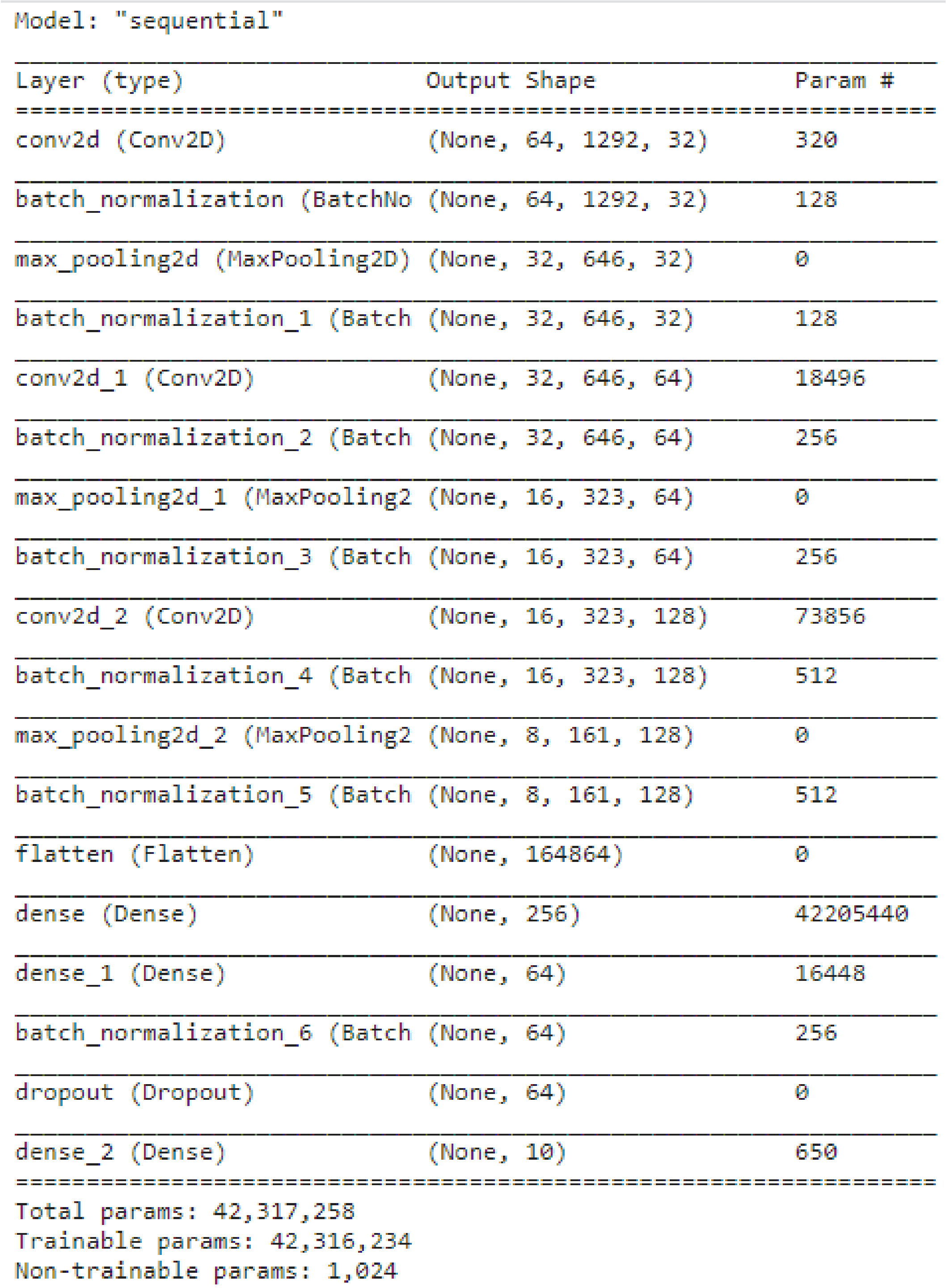
	precision	recall	f1-score	support	Labels
0	0.168224	0.900	0.283465	20.000	Bark
1	1.000000	0.050	0.095238	20.000	Meow
2	1.000000	0.250	0.400000	20.000	Siren
3	0.900000	0.450	0.600000	20.000	Shatter
4	0.800000	0.200	0.320000	20.000	Knock
5	0.750000	0.600	0.666667	20.000	Crying_and_sobbing
6	0.428571	0.450	0.439024	20.000	Microwave_oven
7	0.500000	0.050	0.090909	20.000	Vehicle_horn_and_car_horn_and_honking
8	0.678571	0.950	0.791667	20.000	Doorbell
9	0.600000	0.150	0.240000	20.000	Walk_and_footsteps
accuracy	0.405000	0.405	0.405000	0.405	accuracy
macro avg	0.682537	0.405	0.392697	200.000	macro avg
weighted avg	0.682537	0.405	0.392697	200.000	weighted avg

Figure 11: Various Scores on Validation Set for model 4



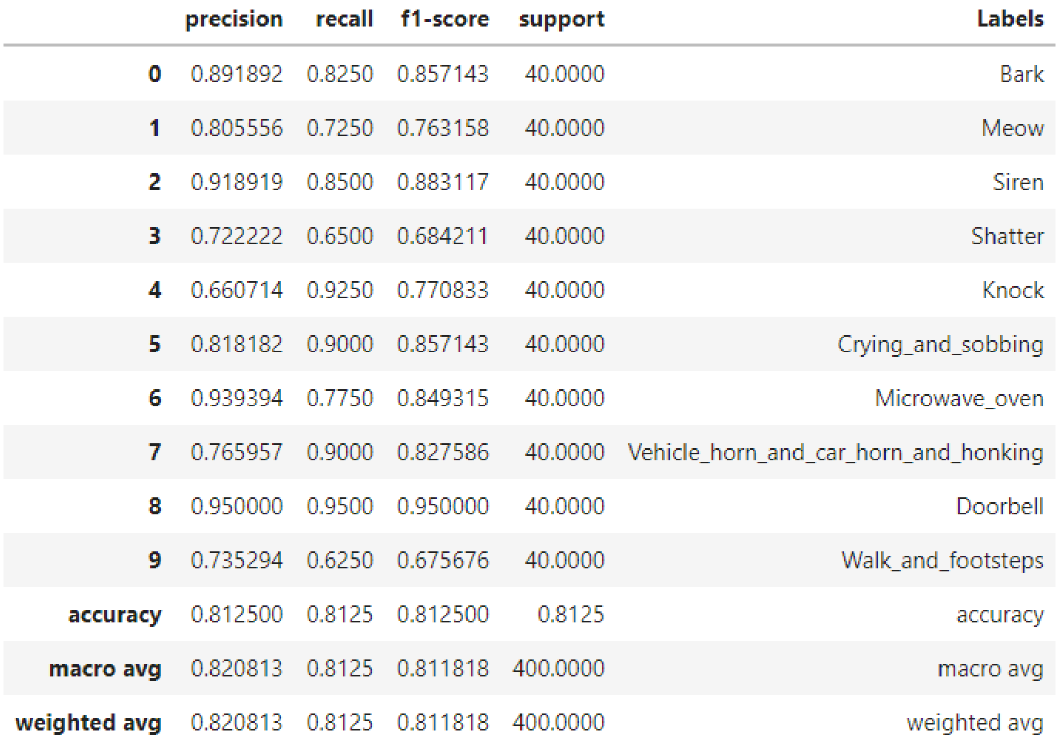
	precision	recall	f1-score	support
Bark	0.434783	0.416667	0.425532	24.00000
Crying_and_sobbing	0.000000	0.000000	0.000000	18.00000
Doorbell	0.625000	0.277778	0.384615	18.00000
Knock	0.000000	0.000000	0.000000	18.00000
Meow	0.000000	0.000000	0.000000	18.00000
Microwave_oven	0.222222	0.555556	0.317460	18.00000
Shatter	0.121951	0.833333	0.212766	18.00000
Siren	0.000000	0.000000	0.000000	27.00000
Vehicle_horn_and_car_horn_and_honking	0.500000	0.041667	0.076923	24.00000
Walk_and_footsteps	0.000000	0.000000	0.000000	18.00000
accuracy	0.203980	0.203980	0.203980	0.20398
macro avg	0.190396	0.212500	0.141730	201.00000
weighted avg	0.198407	0.203980	0.141921	201.00000

Figure 12: Various Scores on Test Set for model 4



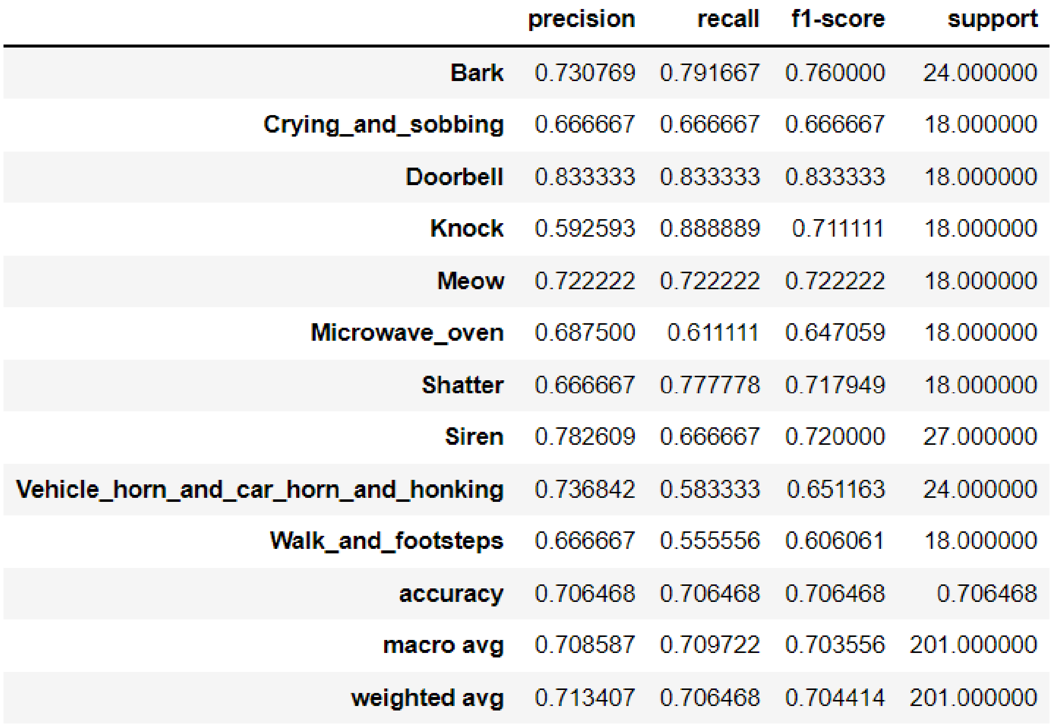
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d (Conv2D)              (None, 64, 1292, 32)      320       
_________________________________________________________________
batch_normalization (BatchNo (None, 64, 1292, 32)      128       
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 32, 646, 32)       0         
_________________________________________________________________
batch_normalization_1 (Batch (None, 32, 646, 32)       128       
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 32, 646, 64)       18496     
_________________________________________________________________
batch_normalization_2 (Batch (None, 32, 646, 64)       256       
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 16, 323, 64)       0         
_________________________________________________________________
batch_normalization_3 (Batch (None, 16, 323, 64)       256       
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 16, 323, 128)      73856     
_________________________________________________________________
batch_normalization_4 (Batch (None, 16, 323, 128)      512       
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 8, 161, 128)       0         
_________________________________________________________________
batch_normalization_5 (Batch (None, 8, 161, 128)       512       
_________________________________________________________________
flatten (Flatten)            (None, 164864)            0         
_________________________________________________________________
dense (Dense)                (None, 256)               42205440  
_________________________________________________________________
dense_1 (Dense)              (None, 64)                16448     
_________________________________________________________________
batch_normalization_6 (Batch (None, 64)                256       
_________________________________________________________________
dropout (Dropout)            (None, 64)                0         
_________________________________________________________________
dense_2 (Dense)              (None, 10)                650       
=================================================================
Total params: 42,317,258
Trainable params: 42,316,234
Non-trainable params: 1,024
```

Figure 13: Model Architecture for model 5



	precision	recall	f1-score	support	Labels
0	0.891892	0.8250	0.857143	40.0000	Bark
1	0.805556	0.7250	0.763158	40.0000	Meow
2	0.918919	0.8500	0.883117	40.0000	Siren
3	0.722222	0.6500	0.684211	40.0000	Shatter
4	0.660714	0.9250	0.770833	40.0000	Knock
5	0.818182	0.9000	0.857143	40.0000	Crying_and_sobbing
6	0.939394	0.7750	0.849315	40.0000	Microwave_oven
7	0.765957	0.9000	0.827586	40.0000	Vehicle_horn_and_car_horn_and_honking
8	0.950000	0.9500	0.950000	40.0000	Doorbell
9	0.735294	0.6250	0.675676	40.0000	Walk_and_footsteps
accuracy	0.812500	0.8125	0.812500	0.8125	accuracy
macro avg	0.820813	0.8125	0.811818	400.0000	macro avg
weighted avg	0.820813	0.8125	0.811818	400.0000	weighted avg

Figure 14: Various Scores on Validation Set for model 5



	precision	recall	f1-score	support
Bark	0.730769	0.791667	0.760000	24.000000
Crying_and_sobbing	0.666667	0.666667	0.666667	18.000000
Doorbell	0.833333	0.833333	0.833333	18.000000
Knock	0.592593	0.888889	0.711111	18.000000
Meow	0.722222	0.722222	0.722222	18.000000
Microwave_oven	0.687500	0.611111	0.647059	18.000000
Shatter	0.666667	0.777778	0.717949	18.000000
Siren	0.782609	0.666667	0.720000	27.000000
Vehicle_horn_and_car_horn_and_honking	0.736842	0.583333	0.651163	24.000000
Walk_and_footsteps	0.666667	0.555556	0.606061	18.000000
accuracy	0.706468	0.706468	0.706468	0.706468
macro avg	0.708587	0.709722	0.703556	201.000000
weighted avg	0.713407	0.706468	0.704414	201.000000

Figure 15: Various Scores on Test Set for model 5

and reduced the training and validation loss very well as for others, it would fluctuate constantly. It achieved an accuracy of about 0.812 which is the best as compared to all other models. But, here also, walk_and_footsteps achieved the lowest f1 score for validation data in figure 14.

Evaluation over test data gave the best accuracy as compared to all other models as shown in figure 15. F1 scores of almost all classes were comparable with doorbell being the best and walk_and_footsteps being the worst.

4 Observations and Discussion:

We observe that out of all these models, data augmentation seems to work best with the applied model being CNN out of all the non-sequential models. In future, for this kind of projects, we can consider data augmentation to increase the number of samples appropriately and it should result in better performance. We observed that it also depends upon the data samples. Some data samples like walk_and_footsteps had bad results for all the models. Meaning that either the samples were highly variable and less in number. Also, mfcc did not seem to improve the performance in this case as the samples are mainly a single event happening in the whole audio. Also, they are probably higher frequency sounds where mfcc would not perform well. Also, for audio classification, CNN along with LSTM and RNN can be tried as a future project, which preserves sequential information also, which can be useful in case of audio data or spectrograms in this case.