Audio Event Detection

Kapil Singla
SPCOM(MSR)
22104405
kapil22@iitk.ac.in

1 Introduction

The goal of Audio event detection is to recognize what is happening in an audio signal . In practice, the goal is to recognize at what temporal instances different sounds are active within an audio signal Sound recognition problem consists of three different stages as pre-processing of signals, extraction of specific features and their classification. In the Signal pre-processing part, it divide the input signal to different segments which is further used for extracting related features. After that for reduction of size of data we use feature extraction and represent the complex data as feature vectors. Clearly it is case of multilabel classification.Therefore, this task addresses two main challenges of i) recognizing an increased number of diverse sound events, and ii) leveraging subsets of training data featuring annotations of varying reliability.

2 Literature Survey

There are various survey papers which have been published recently on the topic. For example an article provide a description covering time and frequency domain features. Regarding speaker recognition, various surveys on features and speaker modeling and in another review they covered types of features and the best-known clustering algorithms in terms of accuracy. This has consequently helped other researchers study existing algorithms and develop innovative algorithms for previously unavailable applications or requirements.

3 Different Methods

3.1 Method - 1 (Reducing data)

- First we have convert the input files into a 3D variable of (10000,64,1000) and passing every function of output file from eventroll to multihot vector function our output gets into a shape of (10000,10).
- Now when we take sum across coloumns of output file we see the following:
  1. 1499 files contains Alarm bell ringing sound.
  2. 993 files contains Blender sound.
  3. 1128 files contains Cat sound.
  4. 2440 files contains Dishes sound
  5. 1412 files contains Dog sound
  6. 1096 files contains Electric shaver toothbrush sound
  7. 1349 files contains Frying sound
  8. 1219 files contains Running water sound
  9. 9201 files contains Speech sound
  10. 1046 files contains Vacuum cleaner sound

- As we can see that maximum occurance file is speech file and rest are all come so it is a case of data imbalancing. We can remove data imbalancing by various methods.
- Here we try to reduce the data so that we can match the data.
- Firstly I have removed those files which contain only speech sound and that were very less in number so it was not affecting.
- So now we take those files which contains only 2 events in which one of the event is speech.
- In this case we we able to reduce the speech file drastically but along with that other files also get reduced so we then make a condition that only reduce maximum of 500 files not more than that.
- After that speech sound came to 3590 and other files were around range of 800-1000.
- So we have reduced some extend of data balancing by not loosing much information from other files.

3.2 Method -2 (Assigning weights)

- In this method for handing the data imbalance we don't do any change in data rather we try to change the loss function for every class by assigning the weights to every part of the binary cross entropy loss function.
- There are different methods for assigning the weights to loss function which are as follows:

1. **Inverse of number of samples(INS)**: Here we weight the samples as the inverse of the class frequency for the class they belong to.
2. **Inverse of square root of number of samples(ISNS)**: Here we weight the samples as the inverse of the Square Root of class frequency for the class they belong to.
3. **Effective number of samples (ENS)**: The re-weighting strategies rely on the total number of samples present in each class. As the number of samples increases, the additional benefit of a newly added data point will diminish. The effective number of samples is defined as the volume of samples.

Table 1: Comparison of different schemes

Part	
Weighting Scheme	F1 score
INS	76.122
ISNS	74.94
ENS	75.561

4 Different Models

4.1 Defining our CNN Model:

4.1.1 First convolution layer

- Having Filters of 16 with a kernal size of 3X3 using activation layer as Relu with Max pooling=2.

4.1.2 Second convolution layer

- Having Filters of 32 with a kernal size of 3X3 using activation layer as Relu with Max pooling=2.

4.1.3 Third convolution layer

- Having Filters of 64 with a kernal size of 3X3 using activation layer as Relu with Max pooling=2.

4.1.4 Fourth convolution layer

- Having Filters of 128 with a kernal size of 3X3 using activation layer as Relu with Max pooling=2.

4.1.5 First hidden layer

- No. of neurons used are 64 with activation layer as Relu.

4.1.6 Output layers

- Output neurons are 10 with activation layer as Sigmoid.

Finally we have used binary categorical cross entropy with optimizer as adam using metrics as precision and recall and a batch size of 32 having 40 epochs.

4.2 Defining LSTM model

4.2.1 First LSTM layer

- Having a parameter of 128 and input shape as 64,1000.

4.2.2 First Dropout layer

- Adding first dropout layer with a parameter of 0.25.

4.2.3 Second LSTM layer

- Having a parameter of 64.

4.2.4 Second Dropout layer

- Adding Second dropout layer with a parameter of 0.25.

4.2.5 First hidden layer

- No. of neurons used are 512 with activation layer as Relu.

4.2.6 Third Dropout layer

- Adding dropout layer with a parameter of 0.25.

4.2.7 Second hidden layer

- No. of neurons used are 64 with activation layer as Relu.

4.2.8 Fourth Dropout layer

- Adding first dropout layer with a parameter of 0.25.

4.3 Evaluating model:

- Now we have use our model for actually testing the test data.
- After predicting the output we have use the round function in the predicted labels and then convert that values into a string
- Now store that labels back into the test.csv file.
- Make classification report which includes precision, recall, F1 score.
- After than we have to find the confusion matrix.

5 Observations

5.1 Testing model on validation data (2000 files)

- Using **CNN** model.

5.1.1 F1 Score without considering the weights in loss function

	precision	recall	f1-score	support
0	0.47	0.20	0.28	292
1	0.71	0.44	0.54	179
2	0.75	0.40	0.52	243
3	0.67	0.41	0.51	479
4	0.65	0.36	0.46	267
5	0.90	0.86	0.88	224
6	0.95	0.84	0.89	269
7	0.83	0.64	0.73	249
8	0.93	1.00	0.96	1852
9	0.91	0.80	0.85	222
micro avg	0.86	0.73	0.79	4276
macro avg	0.78	0.59	0.66	4276
weighted avg	0.82	0.73	0.76	4276
samples avg	0.87	0.74	0.77	4276

Figure 1: F1 score of CNN model without considering weights

5.1.2 F1 Score with the weights in loss function

	precision	recall	f1-score	support
0	0.50	0.48	0.49	292
1	0.66	0.51	0.57	179
2	0.75	0.45	0.56	243
3	0.60	0.52	0.56	479
4	0.59	0.46	0.52	267
5	0.90	0.92	0.91	224
6	0.91	0.90	0.91	269
7	0.86	0.64	0.73	249
8	0.93	0.97	0.95	1852
9	0.93	0.77	0.84	222
micro avg	0.83	0.77	0.80	4276
macro avg	0.76	0.66	0.70	4276
weighted avg	0.82	0.77	0.79	4276
samples avg	0.84	0.78	0.78	4276

Figure 2: F1 score of CNN with the weights in loss function

- Using **LSTM** model

5.1.3 F1 Score without considering the weights in loss function

	precision	recall	f1-score	support
0	0.53	0.36	0.43	392
1	0.46	0.49	0.47	436
2	0.53	0.58	0.55	274
3	0.44	0.20	0.28	444
4	0.48	0.22	0.30	306
5	0.62	0.31	0.42	221
6	0.66	0.48	0.56	130
7	0.23	0.21	0.22	143
8	0.68	0.85	0.76	1246
9	0.30	0.11	0.16	151
micro avg	0.57	0.51	0.54	3743
macro avg	0.49	0.38	0.41	3743
weighted avg	0.55	0.51	0.51	3743
samples avg	0.54	0.50	0.48	3743

Figure 3: F1 score of LSTM model without weights in loss function

5.1.4 F1 Score with the weights in loss function

	precision	recall	f1-score	support
0	0.61	0.30	0.40	392
1	0.51	0.41	0.45	436
2	0.68	0.50	0.57	274
3	0.35	0.29	0.32	444
4	0.46	0.31	0.37	306
5	0.61	0.36	0.45	221
6	0.78	0.31	0.44	130
7	0.28	0.17	0.21	143
8	0.68	0.87	0.76	1246
9	0.34	0.13	0.19	151
micro avg	0.59	0.51	0.55	3743
macro avg	0.53	0.36	0.42	3743
weighted avg	0.57	0.51	0.52	3743
samples avg	0.55	0.50	0.49	3743

Figure 4: F1 score of LSTM model with weights in loss function

5.2 Testing model based on *ACTUAL TEST* samples given (2500 files)

- Using **CNN** model

5.2.1 F1 Score without considering the weights in loss function

	precision	recall	f1-score	support
0	0.50	0.36	0.42	400
1	0.54	0.42	0.47	266
2	0.75	0.41	0.53	284
3	0.71	0.37	0.48	689
4	0.51	0.41	0.45	341
5	0.70	0.48	0.57	283
6	0.78	0.72	0.75	377
7	0.44	0.33	0.38	306
8	0.95	1.00	0.97	2373
9	0.68	0.45	0.54	251
micro avg	0.80	0.68	0.73	5570
macro avg	0.66	0.50	0.56	5570
weighted avg	0.77	0.68	0.71	5570
samples avg	0.81	0.68	0.72	5570

Figure 5: F1 Score of CNN without considering the weights in loss function

5.2.2 F1 Score with the weights in loss function

- *THIS IS COMING OUR BEST CASE FOR THIS PROBLEM*

	precision	recall	f1-score	support
0	0.66	0.48	0.55	400
1	0.38	0.47	0.42	266
2	0.72	0.63	0.67	284
3	0.57	0.40	0.47	689
4	0.49	0.49	0.49	341
5	0.77	0.47	0.58	283
6	0.91	0.66	0.77	377
7	0.62	0.30	0.41	306
8	0.96	0.97	0.96	2373
9	0.65	0.75	0.70	251
micro avg	0.78	0.70	0.74	5570
macro avg	0.67	0.56	0.60	5570
weighted avg	0.77	0.70	0.73	5570
samples avg	0.80	0.72	0.73	5570

Figure 6: F1 Score of CNN with the weights in loss function

- Using LSTM model

5.2.3 F1 Score without considering the weights in loss function

	precision	recall	f1-score	support
0	0.59	0.57	0.58	392
1	0.51	0.55	0.53	436
2	0.69	0.55	0.61	274
3	0.45	0.40	0.42	444
4	0.77	0.37	0.50	306
5	0.68	0.46	0.55	221
6	0.71	0.42	0.52	130
7	0.28	0.19	0.22	143
8	0.67	0.90	0.77	1246
9	0.43	0.30	0.35	151
micro avg	0.61	0.60	0.60	3743
macro avg	0.58	0.47	0.51	3743
weighted avg	0.60	0.60	0.59	3743
samples avg	0.60	0.60	0.56	3743

Figure 7: F1 Score of LSTM without considering the weights in loss function

5.2.4 F1 Score with considering the weights in loss function

	precision	recall	f1-score	support
0	0.65	0.45	0.53	392
1	0.60	0.48	0.53	436
2	0.86	0.44	0.58	274
3	0.52	0.32	0.40	444
4	0.82	0.36	0.50	306
5	0.68	0.44	0.53	221
6	0.72	0.44	0.55	130
7	0.59	0.24	0.34	143
8	0.62	1.00	0.77	1246
9	0.35	0.13	0.19	151
micro avg	0.63	0.59	0.61	3743
macro avg	0.64	0.43	0.49	3743
weighted avg	0.64	0.59	0.58	3743
samples avg	0.63	0.58	0.57	3743

Figure 8: F1 Score of LSTM with considering the weights in loss function

5.3 Confusion matrix

- According to the above F1 tables we can see that CNN model with weights is giving us the best prediction with F1 score of 0.74
- so for that confusion matrix for different classed is :

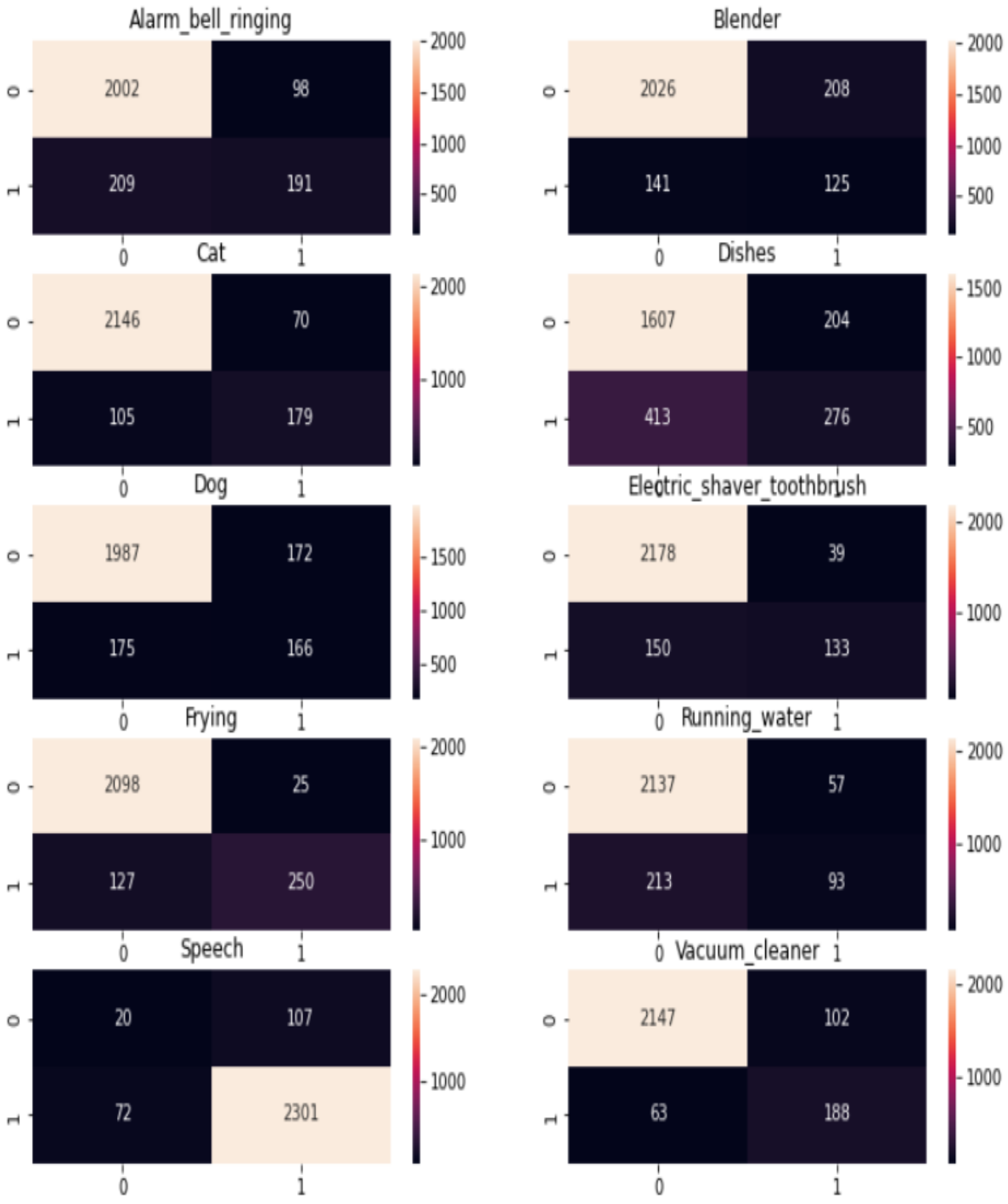

Figure 9: Confusion matrix for CNN with weights

6 Discussion

- **Points learned for handling data Imbalancing :**
- We have used 2 techniques: reducing data and assigning weights to loss function.
- In the first technique the result was not upto the mark may be due to the reduced number of samples so i don't think it is a good approach.
- In the second technique we have got better results in both validation and test data because:
  - We want to weigh the loss computed for different samples differently based on whether they belong to the majority or the minority classes.
  - It basically weighs the contribution of a particular class towards the loss.
  - We essentially want to assign a higher weight to the loss encountered by the samples associated with minor classes.
  - Here we have used Inverse number of samples weighing technique to assign the weights which is giving better results.
- **Points learned for selecting model :**
- We have a Multilabel classification problem and our data set was also not too large due to which CNN proved to be an effective approach.
- When we used the RNN (LSTM) approach on this dataset we have discover a lower accuracy than the the accuracy we can get with the CNN model.
- It was because total number of learnable parameters were less in CNN but very large in LSTM due to which it was not able to learn properly on this small data set.
- On the other hand no doubt LSTM being the slowest to train but if there was more data then LSTM can work much better.
- **Some other observations :**
- It has been seen by splitting the data when we are changing our kernal size , max polling were validation loss starts increasing.This means model is cramming values not learning and model started to overfit.
- Now to decrease overfitting i have used some of the techniques which are as follows:
  1. I have added a dropout layer but it didn't give much better result. Reason for this may be our dataset is not so large.
  2. I have reduce the no. of epochs.

7 References

[1] https://www.youtube.com/watch?v=265-t5HxOR4.

[2]https://medium.com/gumgum-tech/handling-class-imbalance-by-introducing-sample-weighting-in-the-loss-function-3bdebd8203b4.

[3] https://www.youtube.com/watch?v=hraKTseOuJA.