EE603A PROJECT - AUDIO EVENT DETECTION AND SOUND EVENT TAGGING USING CRNN

Arkaprava Biswas

MS-R Student, EE Department
IIT Kanpur 20204407
arkapravab20@iitk.ac.in

Thishyan Raj T

MS-R Student, EE Department
IIT Kanpur 20204420
thishyan20@iitk.ac.in

ABSTRACT

Audio event detection is a growing need in recent years. Here our task is to separate boundaries between music and speech and detect when such event occurs or stops. We have attempted the task using Convolutional Recurrent Neural Network and used ”F1 Metric Loss” as our loss function.

1. INTRODUCTION

There are lots of audios getting generated recent years and a growing need to process them and tag them individually. Music and speech are one of the mostly generated types of audio and its important to detect the sequence of the events with proper timing. Here we will be explaining how we have attempted to solve the task.

2. TRAINING DATASET

For training data, we needed our model to train 10s audio files which contains 3 classes: ”silence”, ”music” and ”speech”. We have generated this data using a full 1 hour long music file and a 1 hour long speech file in .wav format.

The music file consists of a one hour recording of Hindustani Classical music and the speech file consists a one-hour-discussion of six people including one female.

To generate the training dataset, first we have converted both the stereo files into single channel audio files of .wav format. These .wav files are then converted to .npy files. This is because loading a .npy file is much faster than loading the entire audio and directly processing it. Then we have made two segments for each file to create the single class and the both class dataset.
Next for creating 10s audio files, we have done it as follows: Onset and offset events are generated as exponential random variables with validation set onset mean as the distribution

This work was supported by...

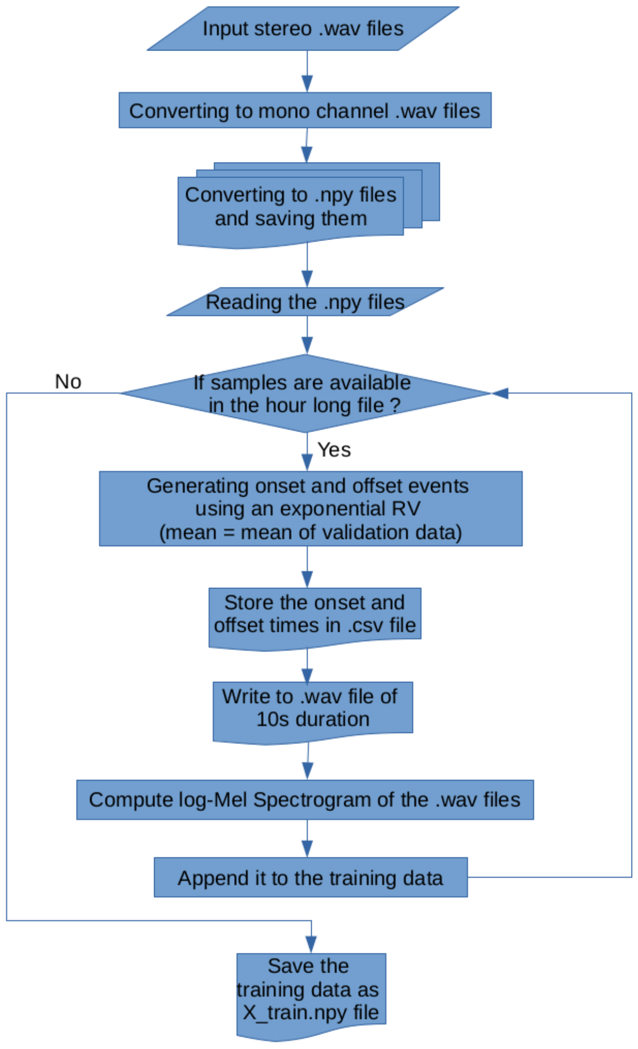

Input stereo .wav files
Converting to mono channel .wav files
Converting to .npy files and saving them
Reading the .npy files
If samples are available in the hour long file ?
No
Yes
Generating onset and offset events using an exponential RV (mean = mean of validation data)
Store the onset and offset times in .csv file
Write to .wav file of 10s duration
Compute log-Mel Spectrogram of the .wav files
Append it to the training data
Save the training data as X_train.npy file

Fig. 1. Flowchart for representing the process of creating the input training data.

mean and the validation set offset mean as the offset distribu-

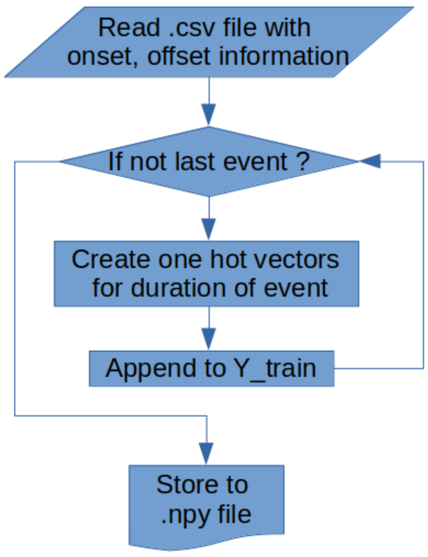

Read .csv file with onset, offset information
If not last event ?
Create one hot vectors for duration of event
Append to Y_train
Store to .npy file

Fig. 2. Flowchart for creating the Output Labels.

tion mean. But it has also been ensured that the event duration is not smaller than 1 second. Then we've sampled from the distributions to create the onset and offset timestamps.
The algorithm followed is : while total audio left for single class not equals zero:
1. Create a zero array of 10s long.
2. Now for each single class file the zeros are replaced by either music or speech defined by the samples from the onset and offset distribution.
3. And for the mixed class files we have taken subsequently from the segments that we have made previously.
This way we have generated the whole batch of 10s audios. The flowcharts for creating the training data and training labels are as shown figure 1 and figure 2.

3. FEATURE SELECTION

For training our model, we have taken log-mel spectrogram of all our 10s samples, and this is the feature that we have taken for training. This is because it is a popularly used feature which has a decent amount of separation between the clusters of Music and Speech. The number of features requiured to capture the information is also smaller compared to STFT. Hence this simplified the architecture of the model and reduced the time required for training. the test data will be The features are of the shape: $\langle Batch, Frequency, time-frames\rangle$.

4. MODEL

We have used Convolutional Recurrent Neural Network model to classify the frames as Speech or Music. The architecture of the CRNN network is as shown in figure 4. Firstly, we have reshaped our training data into into 4 dimensions, as: ¡Batch, Frequency, time-frames, channel¿. Then we've a stack of Convolutional Network, batch normalization, activation and max-pooling. We have done max-pooling across the frequency axis.

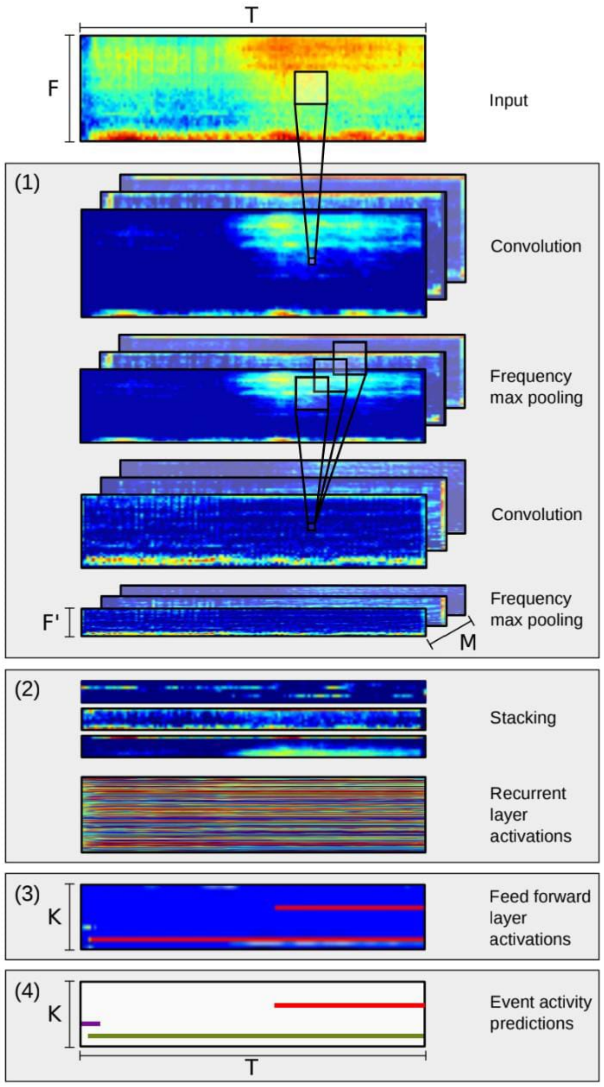


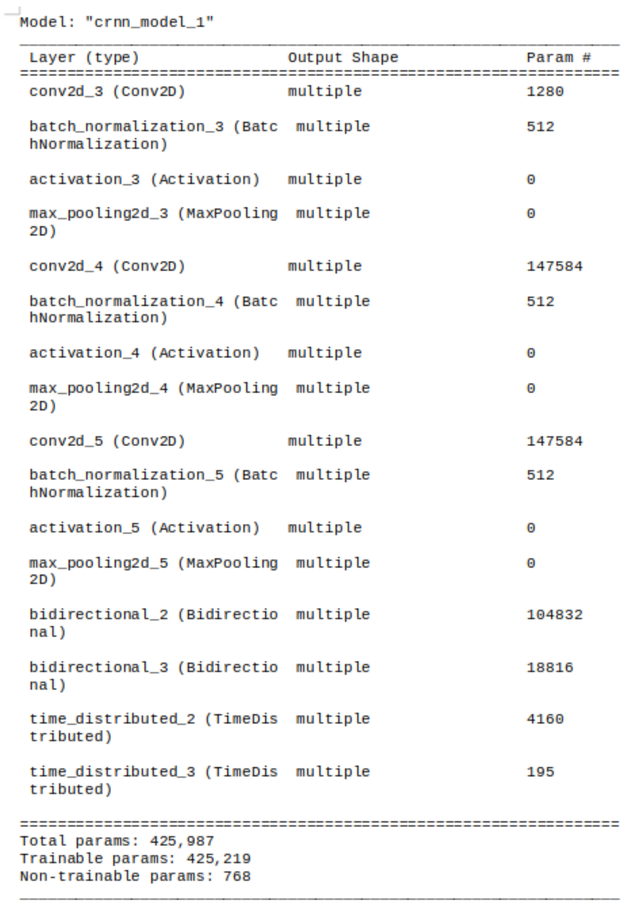
Fig. 3. Architecture of the Convolutional Recurrent Neural Network showing the various layers of the model.

Then we've stacked the frequency and channel together: $\langle Batch, channel * frequency, frames\rangle$.

This will be the input to the RNN layer. The Fully Connected layers are added afterwards and the subsequent time distributed Dense layer to find class probability of each frame.

5. MODEL COMPILATION AND LOSS FUNCTION

We've used Adam optimizer for optimization. And used F1 Metric loss as loss function.
The F1 loss function is basically calculated as:
A classification rule f_b is characterized by a score function $f : X \rightarrow R$ and a threshold $b \in R$ and a classification done according to $f(x) \geq b$.
If Y is the target label and Y^+ is the positive examples and the Y^- are negative examples, then:
True positive = detected and is in annotated.
False positive = not detected but in annotated.
False Negative = detected but not in annotated.
Then the precision and recall is calculated as follows:
Precision = $\frac{TP}{TP+FP}$
And recall = $\frac{TP}{TP+FN}$
And, F1 Score is calculated as = $\frac{2*P*R}{P+R}$
We made it differentiable by taking probabilities instead of hard class assignments. And we minimize $(1 - mean(F1)$ which is same as maximizing F1.



```
Model: "crnn_model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            multiple                  1280

batch_normalization_3 (Batc  multiple                  512
hNormalization)

activation_3 (Activation)    multiple                  0

max_pooling2d_3 (MaxPooling  multiple                  0
2D)

conv2d_4 (Conv2D)            multiple                  147584

batch_normalization_4 (Batc  multiple                  512
hNormalization)

activation_4 (Activation)    multiple                  0

max_pooling2d_4 (MaxPooling  multiple                  0
2D)

conv2d_5 (Conv2D)            multiple                  147584

batch_normalization_5 (Batc  multiple                  512
hNormalization)

activation_5 (Activation)    multiple                  0

max_pooling2d_5 (MaxPooling  multiple                  0
2D)

bidirectional_2 (Bidirectio  multiple                  104832
nal)

bidirectional_3 (Bidirectio  multiple                  18816
nal)

time_distributed_2 (TimeDis  multiple                  4160
tributed)

time_distributed_3 (TimeDis  multiple                  195
tributed)

=================================================================
Total params: 425,987
Trainable params: 425,219
Non-trainable params: 768
_________________________________________________________________
```

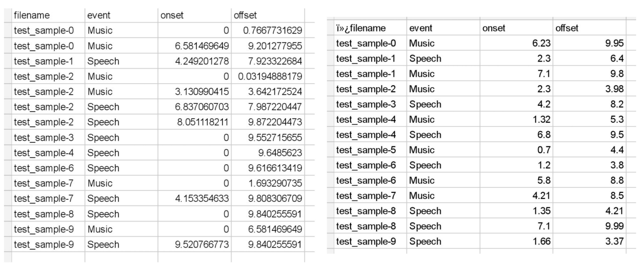
Fig. 4. Model Architecture Parameters of the Convolutional Recurrent Neural Network.

6. RESULTS

Once training is completed the model can be used to predict the events given the test data. Test data is always converted into log-Melspectrogram coefficients and given to the model. The model assigns soft classes to all the frames corresponding to the input data. This is converted to hard classes by performing maximum-value assignment. We then use this to detect the frame number of the events, which is converted into time. Once this is done for all frames of all the input files, this is the required result of the Audio Event Detection task. This is stored for evaluation against the ground truth.

Once we have generated the frame labels for each file, we count the number of frames of each class present in each file. Then we use this information to classify each file as belonging to only Music, only Speech or both Music and Speech files. This is the result required for the second task.

Here is our result for demo test data: For task-1:

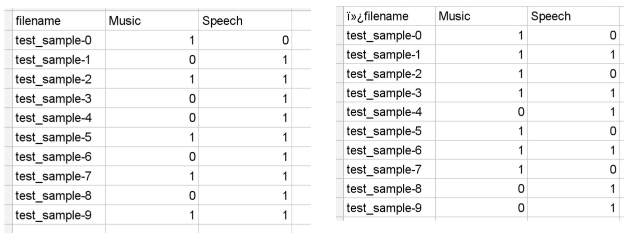


filename	event	onset	offset
test_sample-0	Music	0	0.7667731629
test_sample-0	Music	6.581469649	9.201277955
test_sample-1	Speech	4.249201278	7.923322684
test_sample-2	Music	0	0.03194888179
test_sample-2	Music	3.130990415	3.642172524
test_sample-2	Speech	6.837060703	7.987220447
test_sample-2	Speech	8.051118211	9.872204473
test_sample-3	Speech	0	9.552715655
test_sample-4	Speech	0	9.6485623
test_sample-6	Speech	0	9.616613419
test_sample-7	Music	0	1.693290735
test_sample-7	Speech	4.153354633	9.808306709
test_sample-8	Speech	0	9.840255591
test_sample-9	Music	0	6.581469649
test_sample-9	Speech	9.520766773	9.840255591

ï»¿filename	event	onset	offset
test_sample-0	Music	6.23	9.95
test_sample-1	Speech	2.3	6.4
test_sample-1	Music	7.1	9.8
test_sample-2	Music	2.3	3.98
test_sample-3	Speech	4.2	8.2
test_sample-4	Music	1.32	5.3
test_sample-4	Speech	6.8	9.5
test_sample-5	Music	0.7	4.4
test_sample-6	Speech	1.2	3.8
test_sample-6	Music	5.8	8.8
test_sample-7	Music	4.21	8.5
test_sample-8	Speech	1.35	4.21
test_sample-8	Speech	7.1	9.99
test_sample-9	Speech	1.66	3.37

Fig. 5. Left: Our result in demo test set-1, Right: Ground truth of demo test set-1

For task-2:



filename	Music	Speech
test_sample-0	1	0
test_sample-1	0	1
test_sample-2	1	1
test_sample-3	0	1
test_sample-4	0	1
test_sample-5	1	1
test_sample-6	0	1
test_sample-7	1	1
test_sample-8	0	1
test_sample-9	1	1

ï»¿filename	Music	Speech
test_sample-0	1	0
test_sample-1	1	1
test_sample-2	1	0
test_sample-3	1	1
test_sample-4	0	1
test_sample-5	1	0
test_sample-6	1	1
test_sample-7	1	0
test_sample-8	0	1
test_sample-9	0	1

Fig. 6. Left: Our result in demo test set-2, Right: Ground truth of demo test set-2

7. CONCLUSION AND FUTURE WORK

We're planning to change the hard audio tagging that we've done. We're planning to include a teacher network, that'd take input from the student network which is the frame level probability by instance level attention pooling. And the output of the teacher network will be the bag level probabilities that'd be given input to student network to edit the frame-level-probabilities.

8. REFERENCES

[1] Emre, Giambattista Parascandolo, Toni Heittola, Heikki Huttunen, and Tuomas Virtanen LaTeX*: Convolutional Recurrent Neural Networks for Polyphonic Sound Event Detection*, IEEE Transactions on Audio, Speech and Language Processing, Feb. 2017

[2] Elad Eban, Mariano Schain, Alan Mackey, Ariel Gordon, and Rif A. Saurous LaTeX*:Scalable Learning of Non-Decomposable Objectives*, Google Research, Aug 2016

[3] https://www.youtube.com/watch?v=mA0yDa8UKuE LaTeX*:Speech source for training*

[4] https://www.youtube.com/watch?v=dCMppNvQh4k LaTeX*:Music source for training*