Audio Event Detection

EE698V – Machine Learning for Signal Processing

Shivam Singhal and Ritik Saxena

Aim:

1. Creating a multi-class classifier model to classify 10 audio events.
2. Predicting the sequence of audio-events in merged audio file

Introduction:

In this project, we have to classify the audio event into one of the 10 different events namely – 'air_conditioner','car_horn','children_playing','dog_bark','drilling','engine_idling','gun_shot','jackhammer' ,'siren','street_music'

And then also predict the sequence of audio events from a spectrogram of a merged audio file.

Task 1: Audio Event Classification:

The approach followed by us is as follows:

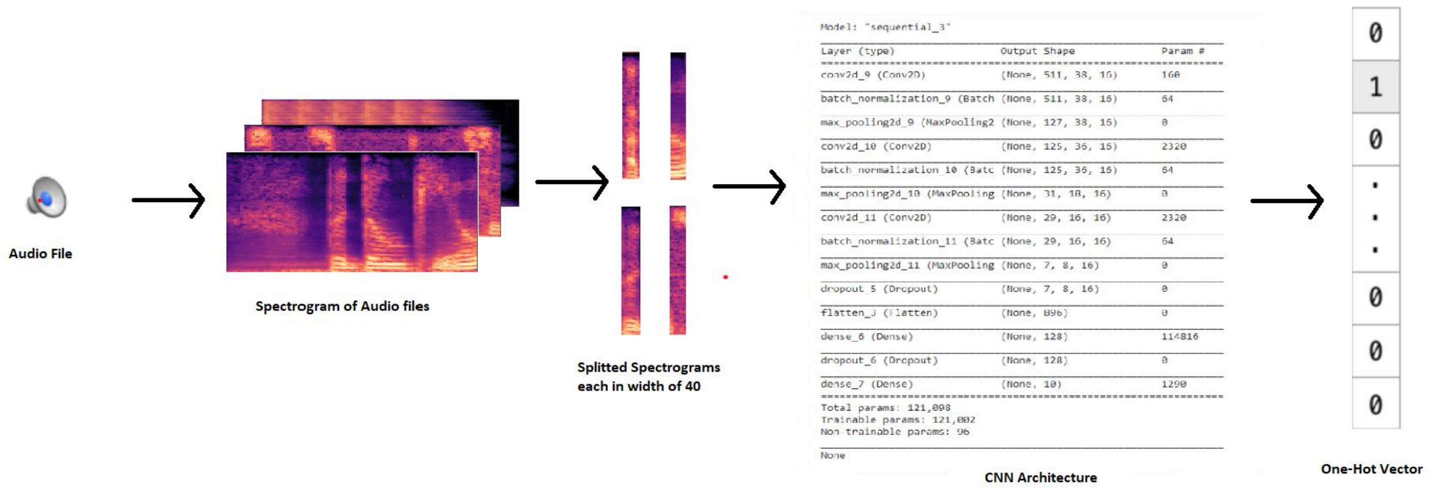
1. Firstly, we removed all repeated audio files so that the model does not overfit.
2. We saw that almost all the temporal information was contained in about 40 frames by hearing audios of each classes.
3. So, we decided to only use 40 frames as input to the neural network.
4. We converted each audio file into spectrogram using librosa library.
5. We split the spectrogram into several files, each file containing 40 frames in time axis.
6. Trained several CNN architectures as we intuitively felt CNN can capture the information well. We did not train the networks for too long to prevent overfitting. We stopped at the point after which the difference in train and dev set accuracy started increasing.
7. Then , we simply applied a majority voting (among all such 40 frame outputs) for predictions on the test set provided.

The best pair of (train accuracy, validation accuracy), we achieved over this dataset is (65,52)

We used a different validation dataset taken from internet sources. **It was however not at all used in training the model.**

Other approach used were to simply fed the CNN network with whole spectrogram but the validation accuracy from that approach was coming out to be only near 30%.

Following is a schematic view of the network:



Audio File
Spectrogram of Audio files
Splitted Spectrograms each in width of 40
CNN Architecture
0
1
0
.
.
.
0
0
0
One-Hot Vector

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_9 (Conv2D)            (None, 511, 38, 16)       160
_________________________________________________________________
batch_normalization_9 (Batch (None, 511, 38, 16)       64
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 127, 38, 16)       0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 125, 36, 16)       2320
_________________________________________________________________
batch_normalization_10 (Batc (None, 125, 36, 16)       64
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 31, 18, 16)        0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 29, 16, 16)        2320
_________________________________________________________________
batch_normalization_11 (Batc (None, 29, 16, 16)        64
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 7, 8, 16)          0
_________________________________________________________________
dropout_5 (Dropout)          (None, 7, 8, 16)          0
_________________________________________________________________
flatten_3 (Flatten)          (None, 896)               0
_________________________________________________________________
dense_6 (Dense)              (None, 128)               114816
_________________________________________________________________
dropout_6 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_7 (Dense)              (None, 10)                1290
=================================================================
Total params: 121,098
Trainable params: 121,002
Non-trainable params: 96
_________________________________________________________________
None
```

Task 2: Audio Event Sequence Detection:

Since the audio file was directly concatenated, we thought to reuse the model trained in Task 1 as a sub routine.

The subroutine gives us labels for every 40 frames.

We then clean the output by removing noise and keeping only the predictions which repeat for sufficient(here, used 3) number of 40 frames. I.e., only those predictions are kept which are surrounded by same prediction.

Then, we finally remove consecutive same labels and give the final predicted sequence of audio events!

For example,

[3,3,3,2,1,3,3,3,5,5,5,5,4,5,5,5,5,7,7,7,7,7,7] is first cleaned to [3,3,5,5,5,5,7,7,7,7]

Then finally [3,3,5,5,5,5,7,7,7,7] is reduced to [3,5,7] and converted to label encoding and given as output.