EE698R: ADVANCED TOPICS IN MACHINE LEARNING

(COURSE PROJECT)

TOPIC: AUDIO EVENT DETECTION

by

Group: ML_questers

Koyel Pramanick	(201333)
Muskan Kaur	(191080)
Sahil Saini	(191118)

Abstract

The objective of this project (Baseline) is to implement CNN models to recognize Audio events from the **UrbanSound9K** dataset. Audio event detection is the task of recognizing the sound events and their respective temporal start and end time in a recording. Sound events in real life do not always occur in isolation but tend to considerably overlap with each other. Using CNN, we observed the accuracy on the test set of 55% which is considerably low, reason being we trained the model using the sample of the dataset having 1500 observations. If we train the same model using bigger data we can get the accuracy of more than 70%. Reason of considering the smaller data is just to decrease the training time.

Introduction

About CNN

CNNS were first used for image classification and recently it got started using on sound related tasks like speech recognition, Audio event detection etc. A conventional CNN consists of convolution layers, pooling layers and fully connected layers. Each convolutional layer consists of a set of learnable kernels. The output of a convolutional layer is a tensor called feature maps. The kernels in a convolutional layer can learn local time-frequency patterns in the spectrogram of an audio clip. In audio processing, low level features can be waveforms or time-frequency representations such as spectrogram. High level features are those extracted from low level features by convolutional layers. Recent CNN architectures apply batch normalization after convolutional layers to speed up and stabilize training. Nonlinear activation functions such as 'ReLU' are applied after each batch normalization. For Sound Event Detection, pooling layers are applied along both time and frequency axes. A time distributed fully connected layer is applied on the output of the last convolutional layer to predict the presence probability of sound events along the time axis. Then the predicted probabilities are aggregated over the time axis to obtain the clip-wise sound event presence probability. The aggregation can be, for example, maximum or average operations over the time axis.

Data Pre-Processing:

We are working on an **UrbanSound8K** datasets. This dataset contains **8732 labeled sound** excerpts (<=4s) of urban sounds from 10 classes. For training purpose, we trained the model using the sample of the dataset having 1600 audio clips.

In addition to the sound excerpts, a CSV file containing metadata about each excerpt is also provided.
Meta-data Files included 8 variables of which, we took only 4 variables, namely audio file name, fold(location), classID and class.

End and start variable are used for selecting uniform sample having length greater than 3s while “slice_file_name”, “fold”, “classID” and class are used to access the audio files by iterating overall dataset.

About a single data point-

- Frequency of each audio clip is divided into 128 components.
- Time divided into 128 frames of 23 ms.

So each data point Xi $\in \mathrm{R}^{128*128}$

The input sound may be longer than 3 seconds so we consider 3 continuous seconds to construct the melspectrogram.

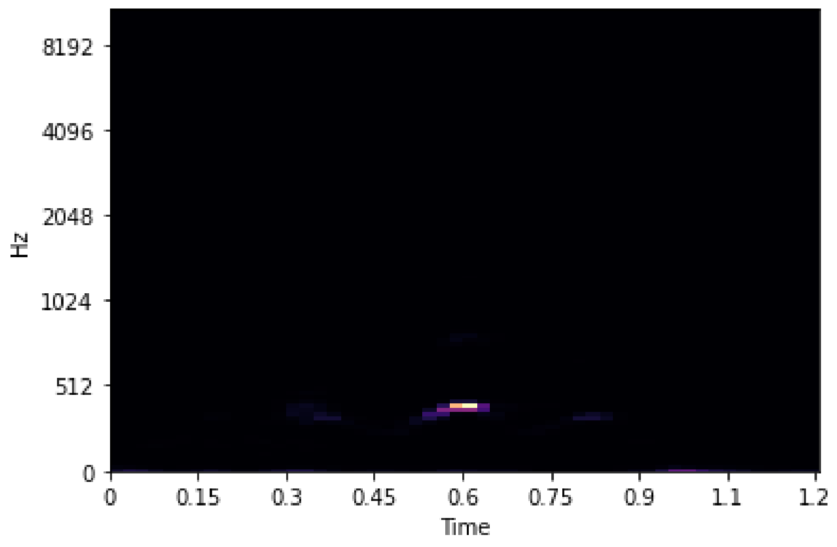
We split the data into training and testing part having 1500 and 100 observations respectively. For inputting the data into CNN framework, we reshaped the data into 128*128*1 and encode the class labels using one hot encoding for 10 classes.

Audio Event Detection Using CNN:

Deep Convolutional Neural Networks (CNN) are, in principle, very well suited to the problem of environmental sound classification as they are capable of capturing energy modulation patterns across time and frequency when applied to spectrogram - like inputs, which has been shown to be an important trait for distinguishing between different, often noise-like, sounds such as children playing and jackhammers . Also, by using convolutional kernels (filters) with a small receptive field, the network should, in principle, be able to successfully learn and later identify spectro-temporal patterns that are representative of different sound classes even if part of the sound is masked (in time/frequency) by other sources (noise).

Now a question comes, what is Spectrogram?

Spectrogram is a visual way of representing the signal strength, or “loudness”, of a signal over time at various frequencies present in a particular waveform. Not only can one see whether there is more or less energy at, for example, 2 Hz vs 10 Hz, but one can also see how energy levels vary over time.



An example of a Mel-spectrogram of Dog Barking

Training the CNN:

In the model, we also used ‘ReLU’ a non-linear activation function in the hidden layers and as we want probabilities of each class in the output layer so ‘Softmax’ activation function is used. We have used ‘Adam’ optimizer and for model the loss function, 'Categorical cross entropy' has been used with 12 epochs.

Visualization of model architecture-

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_3 (Conv2D)            (None, 124, 124, 24)      624       
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 31, 62, 24)        0         
_________________________________________________________________
activation_5 (Activation)    (None, 31, 62, 24)        0         
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 27, 58, 48)        28848     
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 6, 29, 48)         0         
_________________________________________________________________
activation_6 (Activation)    (None, 6, 29, 48)         0         
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 2, 25, 48)         57648     
_________________________________________________________________
activation_7 (Activation)    (None, 2, 25, 48)         0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 2400)              0         
_________________________________________________________________
dropout_2 (Dropout)          (None, 2400)              0         
_________________________________________________________________
dense_2 (Dense)              (None, 64)                153664    
_________________________________________________________________
activation_8 (Activation)    (None, 64)                0         
_________________________________________________________________
dropout_3 (Dropout)          (None, 64)                0         
_________________________________________________________________
dense_3 (Dense)              (None, 10)                650       
_________________________________________________________________
activation_9 (Activation)    (None, 10)                0         
=================================================================
Total params: 241,434
Trainable params: 241,434
Non-trainable params: 0
_________________________________________________________________
```

Performance:

To find the performance of the model, we defined a function which calculates how many samples in the test set got accurately classified divided by total number of samples in the test set.
i.e. (correctly classifies samples/total number of samples)
And we got 56% accuracy on the test set.

Future Work:

Deep neural networks, which have a high model capacity, are particularly dependent on the availability of large quantities of training data in order to learn a non-linear function from input to output that generalizes well and yields high classification accuracy on unseen data.
Next, we will use advance techniques like **Few Shot Learning** to yield better accuracy when we don’t have much data.