EE603: Machine Learning for Signal Processing Assignment 2-Multiple Events Detection

Sonal Agrawal
200995
IIT Kanpur
sagarwal20@iitk.ac.in

1 Introduction

Multiple Audio Event Detection/ Audio Tagging is a research field that aims to detect and identify events in audio signals. In addition to playing a significant role in understanding the audio of real-life sensing, it also has a wide range of applications such as automatic driving, surveillance systems, health care, and humanoid robots. The objective of this project is to build a multi label classifier to identify the sounds that are present from 11 different classes: 'Alarm bell ringing', 'Blender', 'Cat', 'Dishes', 'Dog', 'Electric shaver toothbrush', 'Frying', 'Running water', 'Speech', 'Vacuum cleaner' , 'silence' Concepts such as transfer learning, dealing with data imbalance, CRNN have been implemented and run in this paper

2 Literature Survey

A deep learning CNN consists of three layers: a convolutional layer, a pooling layer and a fully connected (FC) layer. The convolutional layer is the first layer while the FC layer is the last. It's architecture is analogous to the connectivity pattern of the human brain. Just like the brain consists of billions of neurons, CNNs also have neurons arranged in a specific way. Compared to the older networks, a CNN delivers better performance with image inputs, and also with speech or audio signal inputs. Since our data is audio signal I decided to use CNN model for the project. Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task.

3 Input data

3.1 Mel-spectrogram

The dataset contains 10000 spectrograms and their labels, we have been given numpy arrays as inputs that extracted the logarithm of melspectrogram (logMel) of the amplitudes. The dataset contains 10000 spectrograms and their labels. Each spectrograms contains 1 channel, nmels = 64, time frames = 1000

3.2 Audio Dataset

These numpy arrays are of size (64,1000,1). They are stored in the 'X' folder which I extracted in the jupyter notebook using np.load

We have been given a Y folder in dataset which contains the labels, I extracted the names in 'annotations.csv' that contains the names of the 10000 spectrograms along with their labels. Then imported the csv as train_dataset

Table 1: Audio Dataset

Part	
	Size
Train	10000
Validation	2000
Test	2500

4 Method Used

The codes were run on Jupyter Notebook(Files imported locally) as well as Google Colab (files were imported by mounting the google drive to the notebook).

4.1 Importing relevant libraries

Python libraries such as numpy, matplotlib, libros, tensorflow, os, pandas, radom have been used in this code.

4.2 Importing Data

Required files have been added to the current working directory path using os.getcwd(). Pandas and numpy have been used to read csv and load the numpy array respectively.

4.3 Data Visualization

The dataset consists of 10000 event audio spectrograms belonging to multiple of 10 different classes: 'Alarm bell ringing', 'Blender', 'Cat', 'Dishes', 'Dog', 'Electric shaver toothbrush', 'Frying', 'Running water', 'Speech', 'Vacuum cleaner'

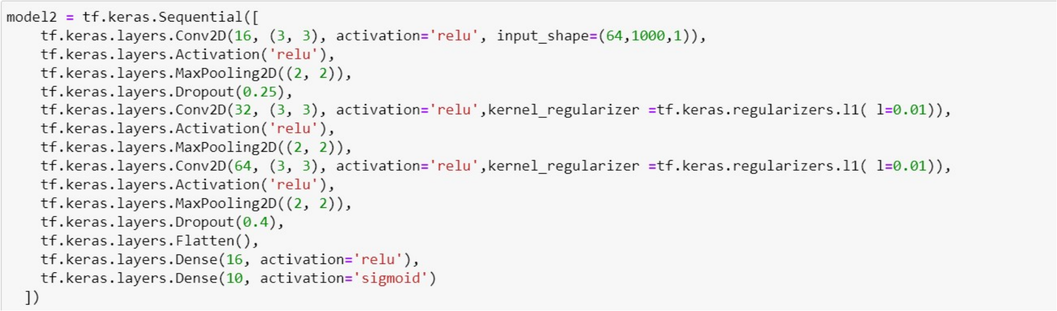
4.4 Converting Y labels to multi hot vectors

Given Y labels had shape of (11,1000) which was converted to (10,) by dropping off the silence class and thus reducing the number of classes from 11 to 10 and also changing the array to multi hot vector under the constraint that if prob>=0.5 consider the label to be present.

4.5 Model : 1

A network using Convolution layers was used to build classifier. The filter_size for each convolution was 3 and number of filters was 16, 32 and 64 for respective layers, activation function used was ReLu

Max pooling(2,2) was used after each convolution layer. During training overfitting was observed, to handle that dropout layers were used with 1 dropout of 0.25 and another of 0.4 and also 'L1' regularization =0.01 was added to both layers. Input shape was fixed as (64,1000,1). Final layer is a dense layer of 10 classes with activation function as sigmoid since it is a multi-label classsifer.



```
model2 = tf.keras.Sequential([
    tf.keras.layers.Conv2D(16, (3, 3), activation='relu', input_shape=(64,1000,1)),
    tf.keras.layers.Activation('relu'),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Dropout(0.25),
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu',kernel_regularizer =tf.keras.regularizers.l1( l=0.01)),
    tf.keras.layers.Activation('relu'),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu',kernel_regularizer =tf.keras.regularizers.l1( l=0.01)),
    tf.keras.layers.Activation('relu'),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Dropout(0.4),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(16, activation='relu'),
    tf.keras.layers.Dense(10, activation='sigmoid')
  ])
```

4.6 TRAINING

4.7 Without considering class weights

Threshold was considered to be 0.5

Train f1 score: 0.5867, Val f1 score: 0.4331

4.8 Considering class weights

The given dataset was highly imbalanced for the given labels. Count of each label =[1499, 993, 1128, 2440, 1412, 1096, 1349, 1219, 9201, 1046]. Thus to deal with the data imbalance, class weights = 1/ frequency were introduced while training the model.
Class weights = [0:0.0006671114, 1:0.0010070493, 2:0.0008865248, 3:0.0004098361 ,4:0.0007082153 , 5:0.0009124088, 6:0.0007412898, 7:0.000820344, 8: 0.0001086838, 9: 0.0009560229]

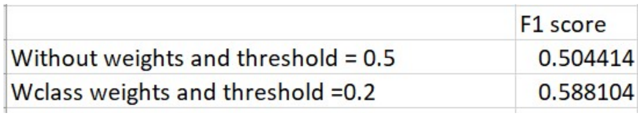
While testing the data the threshold was considered to be 0.2

Train f1 score: 0.5578, Val f1 score: 0.4331

Model : 2

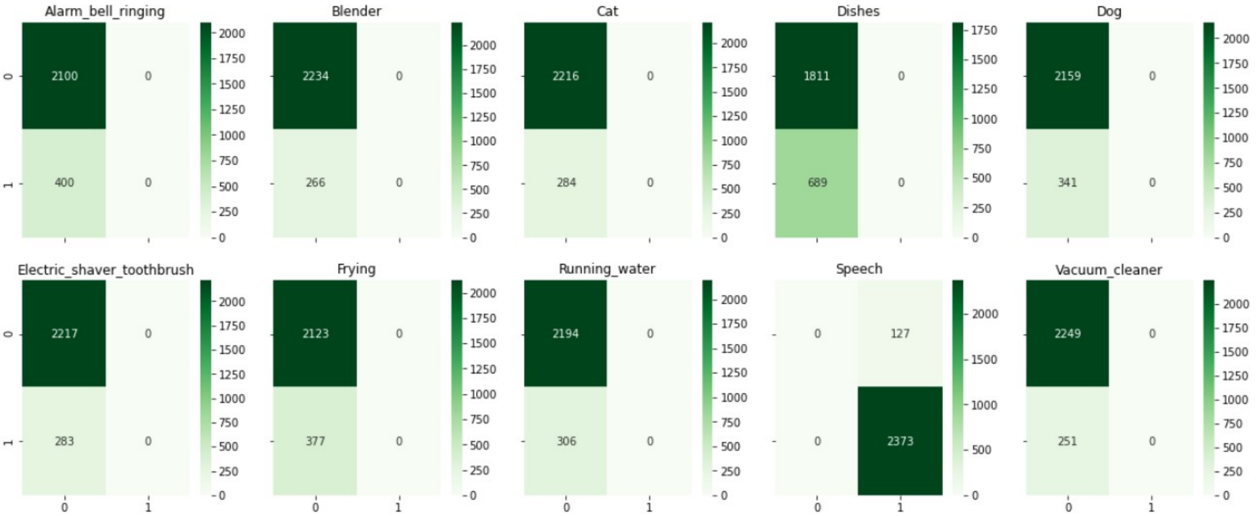
4.9 Final Results

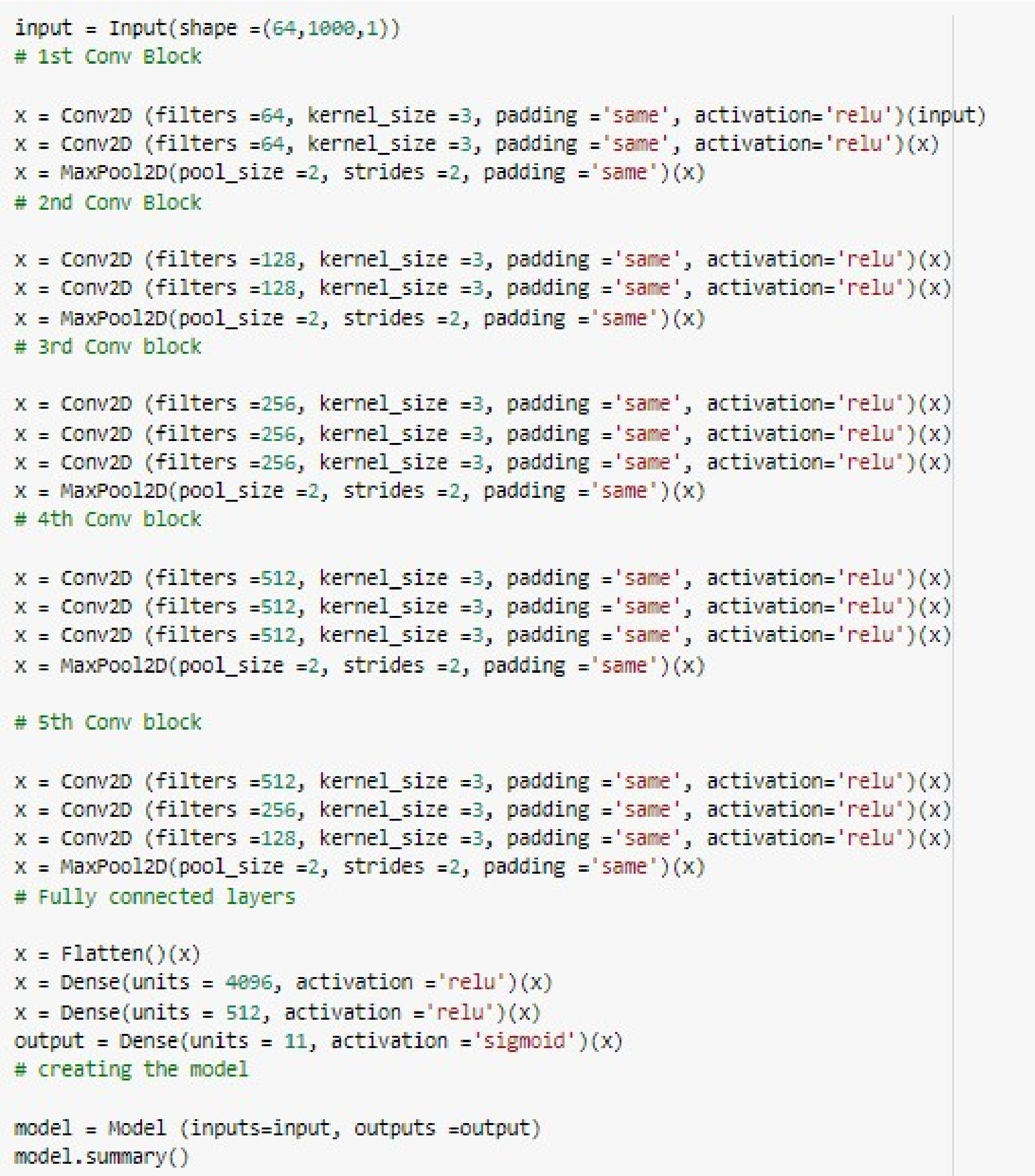
Test Data:



	F1 score
Without weights and threshold = 0.5	0.504414
Wclass weights and threshold =0.2	0.588104

Confusion Matrix:





```
input = Input(shape =(64,1000,1))
# 1st Conv Block

x = Conv2D (filters =64, kernel_size =3, padding ='same', activation='relu')(input)
x = Conv2D (filters =64, kernel_size =3, padding ='same', activation='relu')(x)
x = MaxPool2D(pool_size =2, strides =2, padding ='same')(x)
# 2nd Conv Block

x = Conv2D (filters =128, kernel_size =3, padding ='same', activation='relu')(x)
x = Conv2D (filters =128, kernel_size =3, padding ='same', activation='relu')(x)
x = MaxPool2D(pool_size =2, strides =2, padding ='same')(x)
# 3rd Conv block

x = Conv2D (filters =256, kernel_size =3, padding ='same', activation='relu')(x)
x = Conv2D (filters =256, kernel_size =3, padding ='same', activation='relu')(x)
x = Conv2D (filters =256, kernel_size =3, padding ='same', activation='relu')(x)
x = MaxPool2D(pool_size =2, strides =2, padding ='same')(x)
# 4th Conv block

x = Conv2D (filters =512, kernel_size =3, padding ='same', activation='relu')(x)
x = Conv2D (filters =512, kernel_size =3, padding ='same', activation='relu')(x)
x = Conv2D (filters =512, kernel_size =3, padding ='same', activation='relu')(x)
x = MaxPool2D(pool_size =2, strides =2, padding ='same')(x)

# 5th Conv block

x = Conv2D (filters =512, kernel_size =3, padding ='same', activation='relu')(x)
x = Conv2D (filters =256, kernel_size =3, padding ='same', activation='relu')(x)
x = Conv2D (filters =128, kernel_size =3, padding ='same', activation='relu')(x)
x = MaxPool2D(pool_size =2, strides =2, padding ='same')(x)
# Fully connected layers

x = Flatten()(x)
x = Dense(units = 4096, activation ='relu')(x)
x = Dense(units = 512, activation ='relu')(x)
output = Dense(units = 11, activation ='sigmoid')(x)
# creating the model

model = Model (inputs=input, outputs =output)
model.summary()
```

A Observations and Conclusion

Method with considering class weights and taking threshold gave bettger t=results than threshold at 0.5 Final f1 score = 0.5881. Data imbalance can be treated in a better way and other modesl such as VGG19, RESNET50 and RNN models can also be used. Pytorch can be used rather than tensorflow to deal with data imbalance. It may give better results.