AUDIO EVENT DETECTION AND TAGGING

Technical Report

Shivansh Srivastav (21116007)

Indian Institute of Technlogy, Kanpur
Photonics Science Dept.,
UP 208016, India
shivanshs21@iitk.ac.in

Biswajita Panda (21116401)

Indian Institute of Technlogy, Kanpur
Photonics Science Dept.,
UP 208016, India
biswap21@iitk.ac.in

ABSTRACT

This technical report describes our Acoustic Event Detection and Audio Tagging systems for DCASE2020 challenge Task1[1]. For subtask A, we designed a single model implemented with Neural Networks. We have analyzed a few sample data set created and collected from various devices, and are able to observe 89.9accuracy for the test split. For subtask B, we used the same neural network trained model, to predict the class framewise and observed the onset and offset times for audio event detection.

Index Terms— Neural Networks, Acoustic scene classification, Audio tagging, Audio event detection

1. INTRODUCTION

Sound Classification is one of the most widely used applications in Audio Deep Learning. It involves learning to classify sounds and to predict the category of that sound. This type of problem can be applied to many practical scenarios e.g. classifying music clips to identify the genre of the music, or classifying short utterances by a set of speakers to identify the speaker based on the voice and man more.[2].In this technical report we would like to focus our attention on two main tasks, first being Audio tagging a given audio or spectrogram for the presence of music, speech or both. The second task, takes us a step further to understand the timestamps of the music or speech present in the give audio file.

In this paper we have implemented Audio classification or Tagging by using the TensorFlow machine learning framework.A raw audio dataset has been taken into account and categorized it into 'speech', 'music' and 'music and speech'.This is then followed by pre-processing of data, extracting important features like mel-frequency cepstral components [3], and spectrogram creating, and training a deep learning model to perform classification. This way we can classify incoming audio in one of the three classes.

2. DATASETS

The basic data set was created using some pure music, pure speech, music mixed with noise, speech mixed with noise, and even audio sets containing music, speech along with noise. All in all we compiled 50 such audio sets to train our neural network. Each audio file used was of 10 second long duration and is of format ".wav". The audio data set was sampled at 16000 samples per second to do all the needful computations to extract features. Along with this a csv file was created to keep all the metadata of each audio data set.

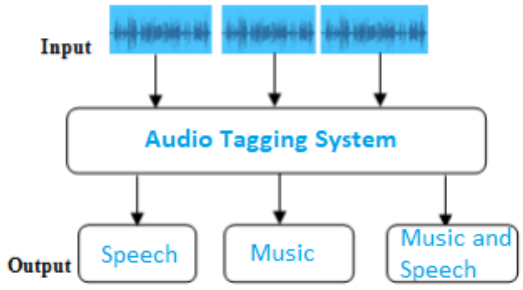

Figure 1: Audio Tagging System

The meta data basically contained the filename and its corresponding label, in other words, this data set corresponds to the ground truth. The validation set which was provided along with the problem statement was further used to test the model which was set up.

3. DATA PREPROCESING

The purpose of pre-processing the data is to study the data so that we can extract features in such a way that they are useful to us to enable their subsequent classification. To simplify this process, we have created a metadata table in csv format containing the filenames and its corresponding labels. This is basically the ground truth table. The audio files are taken and processed using librosa library to calculate its MFCC and spectrograms. Most of the information of any audio file and its behaviour can be extracted by calculating the above mentioned features.

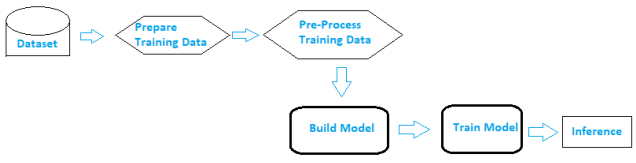

Figure 2: Deep Learning Workflow

Through the above mentioned flowchart, it is an attaempt to explain the flow of creating a dataset, pre-processing it, modeling a neural network, training the neural network, and checking the ac-

curacy of the model by providing it new set of data. The following sections have mentioned a brief discussion on how librosa library, MFCC, and spectrograms extraction, play a crucial role in extracting and classifying audio files.

3.1. Librosa Library

Librosa is a Python package for audio and music signal processing. At a high level, librosa[4] provides implementations of a variety of common functions used throughout the field of music information retrieval. This library hosts a variety of functions like the short time fourier transform, Inverse fourier transforms, functions to load the audio file, display the amplitude values of a sampled audio file, functions to extract features like spectogram, spectral centroid, spectral roll off, spectral bandwidth, zero crossing rate, Mel-Frequency Cepstral Coefficients(MFCCs), Chroma feature and many more such features. In this term project we have particularly used MFCC to train our model. This is done as we are able to extract good results to distunguish between music and speech audio files.

3.2. Mel-Frequency Cepstral Coefficients

The Mel frequency cepstral coefficients (MFCCs) of a signal are a small set of features (usually about 10–20) which concisely describe the overall shape of a spectral envelope. It models the characteristics of the human voice. The MFCC feature extraction technique basically includes windowing the signal, applying the DFT, taking the log of the magnitude, and then warping the frequencies on a Mel scale, followed by applying the inverse DCT.[5]

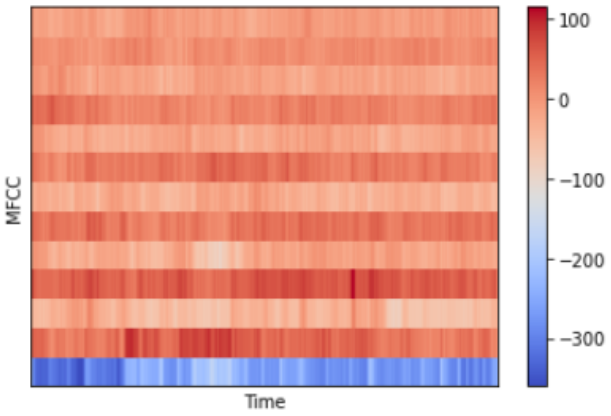

Figure 3: Mel-Frequency Cepstral Coefficients

Mel-frequency cepstral coefficients (MFCCs) are coefficients that collectively make up an MFC.[1] They are derived from a type of cepstral representation of the audio clip (a nonlinear ”spectrum-of-a-spectrum”). The difference between the cepstrum and the mel-frequency cepstrum is that in the MFC, the frequency bands are equally spaced on the mel scale, which approximates the human auditory system’s response more closely than the linearly-spaced frequency bands used in the normal spectrum. This frequency warping can allow for better representation of sound, for example, in audio compression.

MFCCs are commonly derived as follows:

1. Take the Fourier transform of (a windowed excerpt of) a signal.
2. Map the powers of the spectrum obtained above onto the mel scale, using triangular overlapping windows or alternatively, cosine overlapping windows.
3. Take the logs of the powers at each of the mel frequencies.
4. Take the discrete cosine transform of the list of mel log powers, as if it were a signal.
5. The MFCCs are the amplitudes of the resulting spectrum.

There can be variations on this process, for example: differences in the shape or spacing of the windows used to map the scale,[3] or addition of dynamics features such as ”delta” and ”delta-delta” (first- and second-order frame-to-frame difference) coefficients.

3.3. Spectrograms

A spectrogram is a visual representation of the spectrum of frequencies of a signal as it varies with time. When applied to an audio signal, spectrograms are sometimes called sonographs, voiceprints, or voicegrams. Spectrograms are used extensively in the fields of music, linguistics, sonar, radar, speech processing,[1] seismology, and others. Spectrograms of audio can be used to identify spoken words phonetically, and to analyse the various calls of animals. A spectrogram can be generated by an optical spectrometer, a bank of band-pass filters, by Fourier transform or by a wavelet transform (in which case it is also known as a scaleogram or scalogram).

A spectrogram is usually depicted as a heat map, i.e., as an image with the intensity shown by varying the colour or brightness.A common format is a graph with two geometric dimensions: one axis represents time, and the other axis represents frequency; a third dimension indicating the amplitude of a particular frequency at a particular time is represented by the intensity or color of each point in the image.

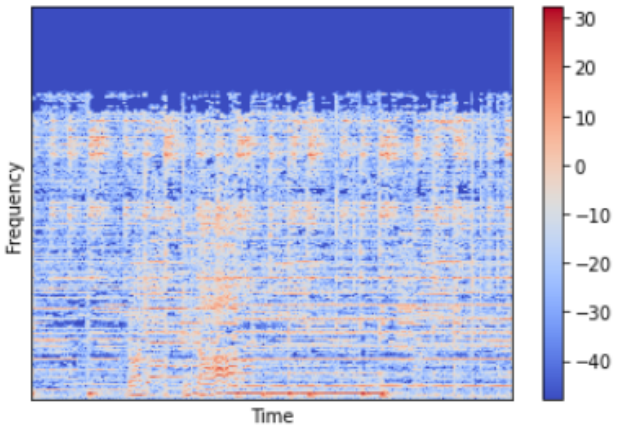

Figure 4: Spectrogram

3.4. Training Data Creation

The audio files are sampled at 16000 samples per second and their MFCCs are calculated and stored in a numpy array. The program

uses 40 MFCC coefficients froptimal results and the final dimensions of MFCC array is (40,313). In order to reduce the dimension of the extracted features, mean is taken for each row of the array to obtain MFCC mean in a one dimensional format. This way the independent variable (x) which is the MFCC feature corresponding to each file is generated. Subsequently, the labels are also extracted for each feature from the ground truth table, this forms our dependent variable (y). The features and label are compiled in a dataframe. This completes our dataset creation.

Going forward, the dataset is then split into 80:20 ratio randomly, for training and validation dataset respectively. This training data is stored in x.train and testing data or the data that will be used for the validation of the model implemented is stored in x.test. Similarly, the labels of 'music', 'speech', and 'speech and music' are encoded into one-hot vector as shown below.

- music : [1 0]
- speech : [0 1]
- music and speech :[1 1]

The independent vector that is y.train consists the encoded labels of ground truth table. This way the training data creation is complete.When the training of model is being done, the data loader will randomly fetch one batch of input features containing the list of audio file names and run the pre-processing audio transforms on each audio file. It will also fetch a batch of the corresponding target labels. Thus it will output one batch of training data at a time, which can directly be fed as input to our deep learning model.

4. MODEL CREATION AND TRAINING

The model that is attempted to impement in this term project is an artificial neural network. [6] Neural Networks is a machine learning technique which is modeled based on the structure of human brain. It consists of networks of learning units called neurons. An artificial neural network is composed of neurons with learning weights and bias. Each neuron receives several inputs, takes over a weighted sum of input and passes through activation function to generate an output. NNs are designed to process multiple arrays of data. It includes 1D for signals and sequences, 2D for images and audio spectrograms, and 3D for video data.NN uses local links, weights, pooling and multilayer usage as the key properties from any signal. A typical NN is organized as a number of stages.

Here the Sequential model of keras is used. The sequential model is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor.A Sequential model can be created by passing a list of layers to the Sequential constructor. Generally, all layers in Keras need to know the shape of their inputs in order to be able to create their weights. So while creating a layer , initially, it has no weights.Th weights are created for the first time on an input, as the shape of the weights depends on the shape of the inputs. Here, the inputs given are 40 which are the mfcc 40 coefficients. The result of this weighted sum value is then passed through a non linear activation function called ReLU.The 'ReLu' activation is used in all layers with 0.5 dropout ratio. A fully connected layer learns the weights, number of weights, and activation function. Other hyper parameters required to fine tune the ANN includes optimizer, learning rate, loss function, mini batch size, epochs, regularization, and weight initialization.

The **learning rate** is defined in order to optimize and minimize the loss function of a ANN model and controls how much the optimization algorithm needs to update its weights. Depending on the choice of the optimizer, the learning rate can be either fixed or varied. Different optimizers are available in CNN which includes Stochastic Gradient descent (SGD), Adam, Adagrad, AdaDelta or RMSProp. Usually the default learning rate works well with Adam optimizer.

Batch size is a term used in machine learning which describes the number of samples included for training in single iteration. The standard batch sizes used are 8, 16, 32, or 64. If the batch size is selected too small, then gradient descent will not be smooth which will makes the model to learn slowly and there may be a high loss. If the batch size is too high then it may take long time to train.

Dropout is a regularization technique used to prevent over fitting of the deep learning model. This method is used to drop some range of units in neural network according to desired ratio. A fully connected layer takes most of parameters and therefore during training, the neurons acquire interdependency with each other which limits the power of each neuron may lead to over fitting of training data.

The proposed ANN model for audio tagging was developed on Google Colab. Colaboratory is a cloud service based on jupyter notebook for machine learning research which improves working with deep learning libraries like PyTorch, Keras, and TENSORFLOW. Colab supports Graphics Processing Units (GPU) as a backend for free. It is used as a tool for accelerating deep learning applications,threby making it suitable for our project.

5. INFERENCE

Inference basicaly means testing the model on new data and finding the accuracy with which it predicts the label. As part of the training loop, it is generally seen that evaluation of metrics is done on the validation data. Post that the inference is done on on unseen data, perhaps by keeping aside a test dataset from the original data. However, for the purposes of this project, we have used the validation data provided with the problem statement for this purpose.The forward pass is executed with the model to get predictions for the various dataset available in the validation dataset. It is observed that the model is running with 89.9 percent accuracy on the dataset. This gives us a reasonable result for the limited data which was used for a fairly basic neural network model.

The task 2 of the same problem statement is an extension of the same model, which is completed by using the librosa functions of onset and offset to find the start and end time of speech or music.

The term project offered another challenge of providing spectrograms instead of audio files to do the audio tagging task. The spectrograms given were converted into audio files with some phase distortion present as the spectrograms consists of absolute value of the audio amplitude spectrum. This simple tweak enabled the prediction of labels for spectrograms provided.

6. CONCLUSION

The general purpose audio tagging system based on deep learning model can be effectively used in various audio based information retrievals. The large scale data used for analysis contains subsets of training data with imbalanced annotations of varying reliability and variable length of audio files. To handle the large scale data, Google Colab cloud service with GPU has been used which improved the learning speed of the models. A baseline system on ANN model is performed using MFCC as the basis feature of the audio file with

a average accuracy of 0.8 with almost 500 seconds of training data (50 audio files of 10 second long duration).

7. REFERENCES

[1] http://dcase.community/challenge2020/.

[2] https://towardsdatascience.com/audio-deep-learning-made-simple-sound-classification-step-by-step-cebc936bbe5.

[3] M. Hossan, S. Memon, and M. Gregory, "A novel approach for mfcc feature extraction," 01 2011, pp. 1 – 5.

[4] C. R. D. L. D. P. E. M. M. E. B. McFee, Brian and O. Nieto, "librosa: Audio and music signal analysis in python," 01 2015, pp. 18 – 25.

[5] PDF specification for Springer, https://link.springer.com/content/pdf/bbm%3A978-3-319-49220-9%2F1.pdf.

[6] S. Walczak and N. Cerpa, "Artificial neural networks," in *Encyclopedia of Physical Science and Technology (Third Edition)*, third edition ed., R. A. Meyers, Ed. New York: Academic Press, 2003, pp. 631–645. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B0122274105008371