Audio Classification Device Using Deep Learning Model and UrbanSound8K dataset

Yash Raj
Electrical and Electronics Department
2001EE83
Indian Institute of TechnologyPatna
Patna, Bihar
yashrj2083@gmail.com

Udit Sethi
Electrical and Electronics Department
2001EE93
Indian Institute of Technolog Patna
Patna, Bihar
sethiudit2000@gmail.com

Abstract— Audio classification or sound classification can be referred to as the process of analysing audio recordings. This amazing technique has multiple applications in the field of AI and data science such as chatbots, automated voice translators, virtual assistants, music genre identification, and text to speech applications. Audio classifications still rely on numerous representations of the audio signal together with large architectures, fine-tuned from large datasets. By utilizing the inherited lightweight nature of audio and novel audio augmentations, we were able to present an efficient end-to-end (e2e) network with strong generalization ability. Experiments on a variety of sound classification sets demonstrate the effectiveness and robustness of our approach, by achieving state-of-the-art results in various settings.

I. INTRODUCTION

Audio Classification is a machine learning task that involves identifying and tagging audio signals into different classes or categories. The goal of audio classification is to enable machines to automatically recognize and distinguish between different types of audio, such as dog bark, gun shot, car horn, etc. We will be implementing Audio classification by using the TensorFlow machine learning framework. We would be considering an audio dataset and categorized it into different classes. Followed by pre-processing, creating, and training a deep learning model to perform classification. This way we can classify incoming audio in one of the two classes, in either speech or music. One of the most widely used applications in Deep Learning is Audio classification, in which the model learns to classify sounds based on audio features. When given the input, it will predict the label for that audio feature. These can be used in different industrial applications like classifying short utterances of the speakers, music genre classification, etc. In this guide, we will walk through a simple demonstration of audio classification on the UrbanSound8K dataset. We will write certain audio pre-processing functions, we will extract Mel Spectrogram from audio, then pass the features to a basic convolutional neural network and start the training, specifically focusing on the classification of particular urban sounds. Our goal is to give an audio file as the input, then our model should determine whether the audio features contain one of the target labels.

II. Components Used

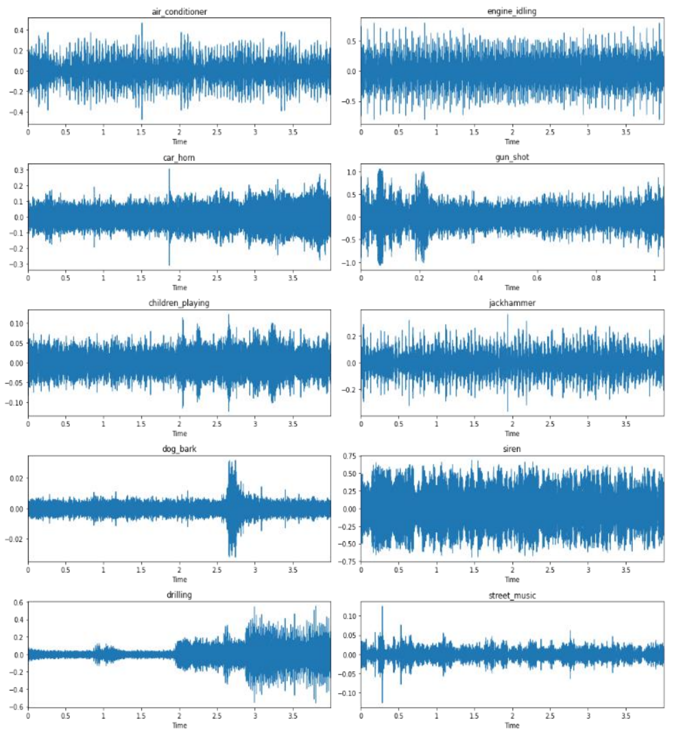
- Raspberry Pi: -

- Servo Motors

- Led Lights

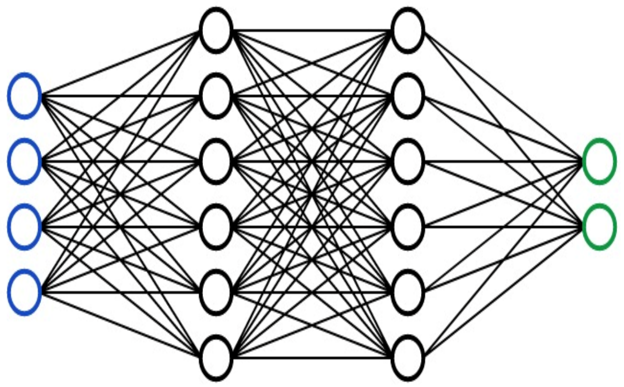
III. Data Set

UrbanSound8K is an audio dataset containing 8732 labeled sound samples, belonging to 10 class labels, split to 10 folds, that is air_conditioner, car_horn, children_playing, dog_bark, drilling, enginge_idling, gun_shot, jackhammer, siren, and street_music. It also contained a CSV file UrbanSound8k.csv which contains meta-data information about every audio file in the dataset. The samples last up to 4 seconds and the sampling rate varies from 16KHz-48KHz. The classes are drawn from the urban sound taxonomy and all excerpts are taken from field recordings 2 . The experiment was conducted on the official 10 fold split, with samples resampled to 22.05KHz and zero-padding the short samples to 4 seconds.



IV. Model Training

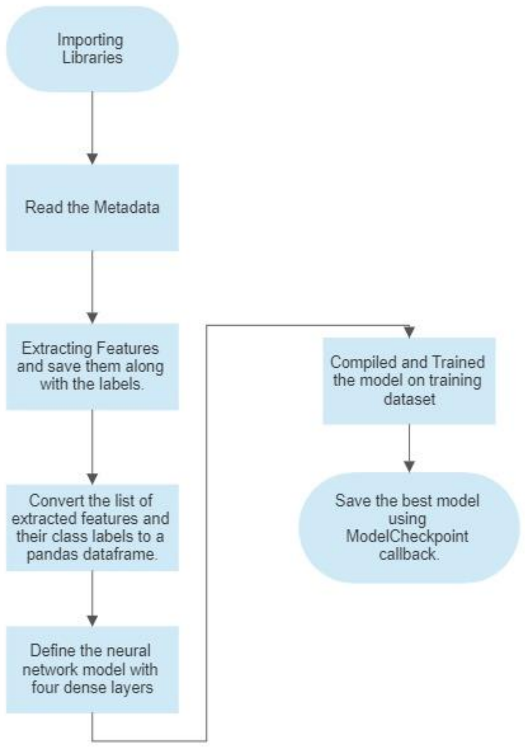
The code is a Python script that performs audio classification on the UrbanSound8K dataset using a neural network model.



The code starts by importing the necessary libraries for data manipulation and feature extraction, such as pandas, numpy, and librosa. Then, it reads the metadata file of the UrbanSound8K dataset and performs some data exploration, checking the balance or imbalance of the dataset. After that, the code defines a function to extract the Mel-frequency cepstral coefficients (MFCCs) from each audio file of the dataset. The MFCCs are a representation of the spectral envelope of an audio signal, commonly used in speech and music processing applications.
Next, the script extracts the MFCCs features from all the audio files of the dataset and stores them in a Pandas dataframe. The dataframe is saved to a CSV file for future use. Then, the features are split into training and testing sets and the labels are one-hot encoded using the to_categorical() function from the Keras library. The neural network model is then defined, consisting of several dense layers with dropout regularization and ReLU activation, and a final softmax layer with the number

of output classes corresponding to the number of unique labels in the dataset. The model is compiled using the categorical_crossentropy loss function, the Adam optimizer, and the accuracy metric.

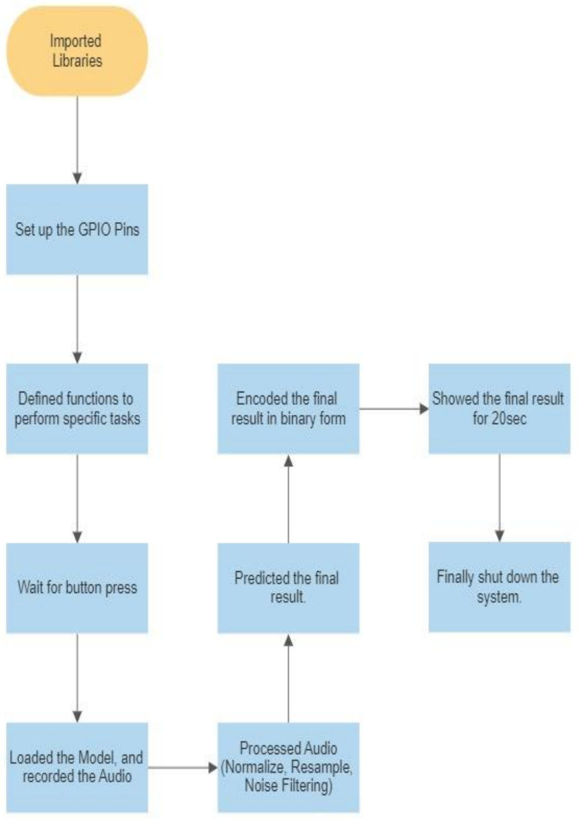
Finally, the model is trained on the training set for a fixed number of epochs, with early stopping and model checkpoint callbacks to prevent overfitting and save the best model weights. The training progress is monitored and the results are printed to the console. The model is evaluated on the testing set using various metrics such as accuracy, precision, recall, and F1 score.


Importing Libraries
Read the Metadata
Extracting Features and save them along with the labels.
Convert the list of extracted features and their class labels to a pandas dataframe.
Define the neural network model with four dense layers
Compiled and Trained the model on training dataset
Save the best model using ModelCheckpoint callback.

V. Hardware Implementation

Here are some of the main steps that are happening in the hardware implementation code:

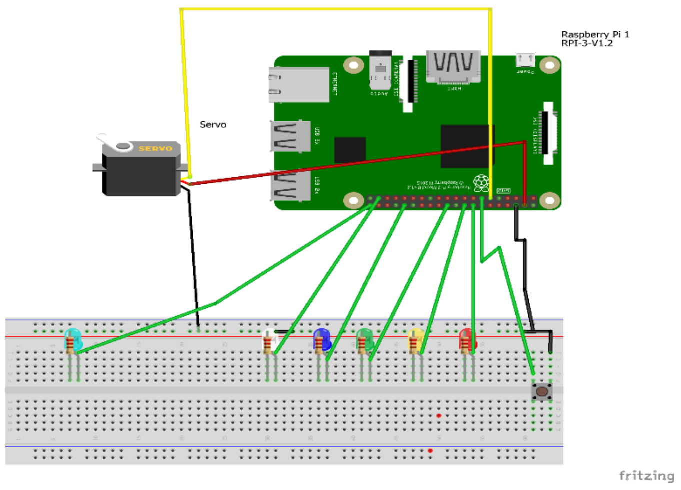
- ➢ Importing necessary libraries including librosa, numpy, pandas, and RPi, GPIO.
- ➢ Setting up the GPIO pins to control the LEDs.
- ➢ Defining functions to turn the LEDs on and off, as well as a function to blink one of the LEDs.
- ➢ Loading a trained TensorFlow Lite model from a file.
- ➢ Loading a metadata CSV file and extracting the class names from it.
- ➢ Recording a 5-second audio file using the arecord command.
- ➢ Starting a thread to blink an LED while the audio is being recorded.
- ➢ Processing the recorded audio file using various techniques including applying a high-pass filter, spectral subtraction, normalization, resampling, STFT, and MFCCs.
- ➢ Making a prediction on the processed audio using the model trained.


Imported Libraries
Set up the GPIO Pins
Defined functions to perform specific tasks
Wait for button press
Loaded the Model, and recorded the Audio
Processed Audio (Normalize, Resample, Noise Filtering)
Predicted the final result.
Encoded the final result in binary form
Showed the final result for 20sec
Finally shut down the system.

```
Total params: 1,670,810
Trainable params: 1,670,810
Non-trainable params: 0
_________________________________________________________________
```

- Turning on the LED associated with the predicted class.

Implementing this on raspberry pi and connecting the circuit according to the described way



VI. Results

Our model determines whether the audio features contain one of the target labels.

We recorded the audio signal and after processing it using different audio processing techniques like sampling and noise filtering, we got the final processed audio and then determined the output using binary encoding of 4-bit using LEDs.
The encoding which we use is given below:

# 1 = car_horn
# 2 = children_playing
# 3 = dog_bark
# 4 = drilling
# 5 = engine_idling
# 6 = gun_shot
# 7 = jackhammer
# 8 = siren
# 9 = street_music
# 15 = air_conditioner

VII. References

[1] Neural Networks for Pattern Recognition, Christopher M. Bishop,1995

[2] A. L. Brown, J. Kang, and T. Gjestland. Towards standardization in soundscape preference assessment. Applied Acoustics, 72(6):387–392, 2011.

[3] L.-H. Cai, L. Lu, A. Hanjalic, H.-J. Zhang, and L.-H. Cai. A flexible framework for key audio effects detection and auditory context inference. IEEE TASLP, 14(3):1026–1039, 2006. 2009.

[4] https://www.analyticsvidhya.com/blog/2021/06/introduction-to-audio-classification/

[5] https://paperswithcode.com/task/audio-classification

[6]https://www.freecodecamp.org/news/beginners-guide-to-raspberry-pi-6e55080fdaaf/

[7] https://en.wikipedia.org/wiki/Neural_network

[8] https://urbansounddataset.weebly.com/urbansound8k.html