Configuration Manual

MSc Research Project
MSc in Data Analytics

Raunak Milind Sathe
Student ID: x20118350

School of Computing
National College of Ireland

Supervisor: Michael Bradford

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Raunak Milind Sathe		
Student ID:	x20118350		
Programme:	MSc in Data Analytics	**Year:**	2021-2022
Module:	Research Project		
Lecturer:	Michael Bradford		
Submission Due Date:	31/01/2022		
Project Title:	Impact of Missing Data on Audio Genre Classification using Convolutional Neural Network		
Word Count:	855		
Page count:	9		

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Raunak Milind Sathe

Date: 31/01/2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Raunak Milind Sathe

x20118350

This manual is a guide to implement the project from scratch. All the steps including environmental setup to running the code are explained in this manual. This manual also contains a section providing a reference to the code source.

1. Hardware Requirements –

The following are the hardware requirements of the machine that the project was run on. It is highly recommended that a minimum of these specifications are met to run the project.

- Processor - Intel I5
- Ram – 4 GB
- Operating System – Windows 10
- Hard disk – 1 TB
- Graphics card – Nvidia GeForce MX150.

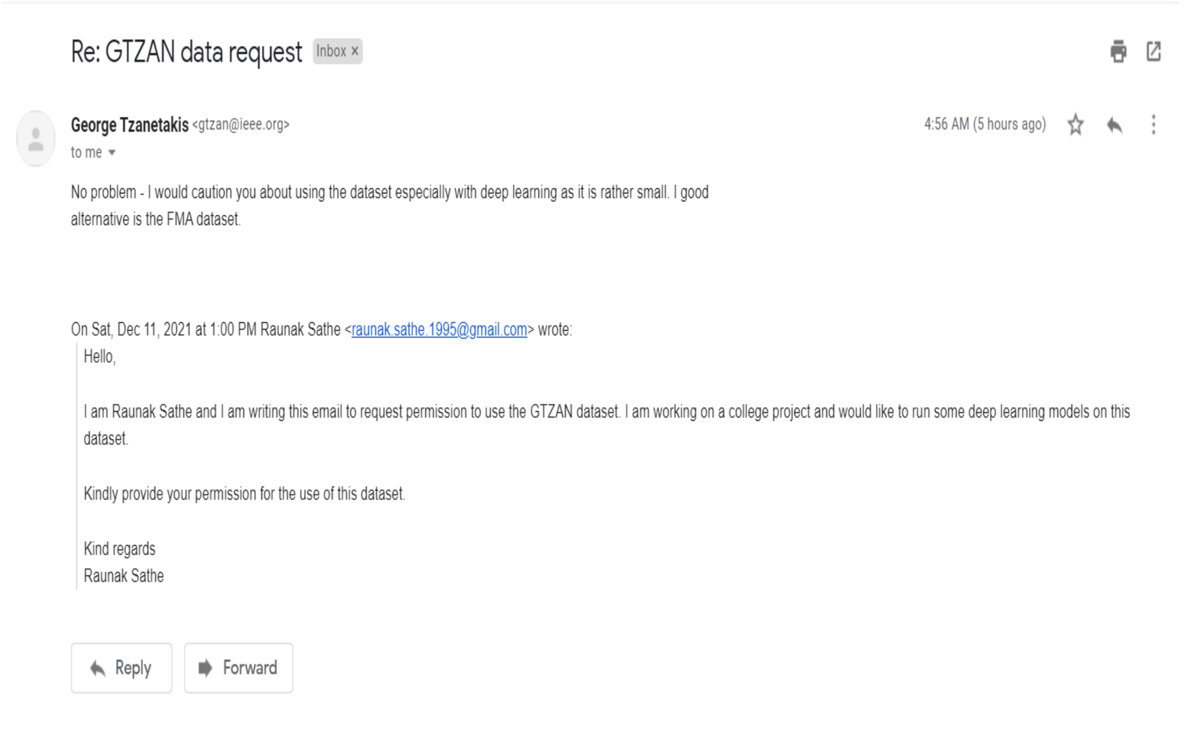
2. Software Requirements –

The following are the software requirements for the implementation of the project. This includes software tools, packages, environments etc.

- Google Drive – The dataset will stored in the Google Drive
- Google Colaboratory – This is a product developed by Google Research, which is a cloud-based version of the Jupyter notebook. This product allows for the running of the heavy machine learning and deep learning code easily that otherwise would be difficult to run on a machine with low specifications.
- Python 3
- Overleaf

3. Data Acquisition –

The dataset was acquired from the Marsyas website. The dataset is a public dataset with no copyright permissions and no titles. However, the creator has asked for permission for the use of the dataset. The following screenshot shows an email conversation with the creator Mr. George Tzanetakis for the permission to use the GTZAN dataset -



Re: GTZAN data request Inbox ×

George Tzanetakis <gtzan@ieee.org>
to me
4:56 AM (5 hours ago)

No problem - I would caution you about using the dataset especially with deep learning as it is rather small. I good alternative is the FMA dataset.

On Sat, Dec 11, 2021 at 1:00 PM Raunak Sathe <raunak.sathe.1995@gmail.com> wrote:

Hello,

I am Raunak Sathe and I am writing this email to request permission to use the GTZAN dataset. I am working on a college project and would like to run some deep learning models on this dataset.

Kindly provide your permission for the use of this dataset.

Kind regards
Raunak Sathe

Reply Forward

4. Code Reference –

The code used in this project has been sourced from external resources. There are four resources referenced for building the code which has been modified as per requirements.

- The data pre-processing – this code has been referenced from the following website – https://hackernoon.com/audio-handling-basics-how-to-process-audio-files-using-python-cli-jo283u3y
- Spectogram generation and image augmentation – this code covers the first 6 cells in both the notebooks. This code has been referenced from - https://github.com/nageshsinghc4/Audio-Data-Analysis-Using-Deep-Learning/blob/master/Audio-Data-Analysis-CNN.py. This section has again been referenced for 2 more cells from the 13th and 14th cells
- The VGG16 model code which covers the next 6 cells in the code have been referenced from the following resource - https://github.com/krishnaik06/Transfer-Learning/blob/master/face_Recognition.py
- The final 3 cells of the code have been referenced from the following resource - https://colab.research.google.com/drive/1-RNHrPU4c_o0-mqhM82Cx428CCwCFrqR#scrollTo=2L17wQvhmC43.

5. Implementation –

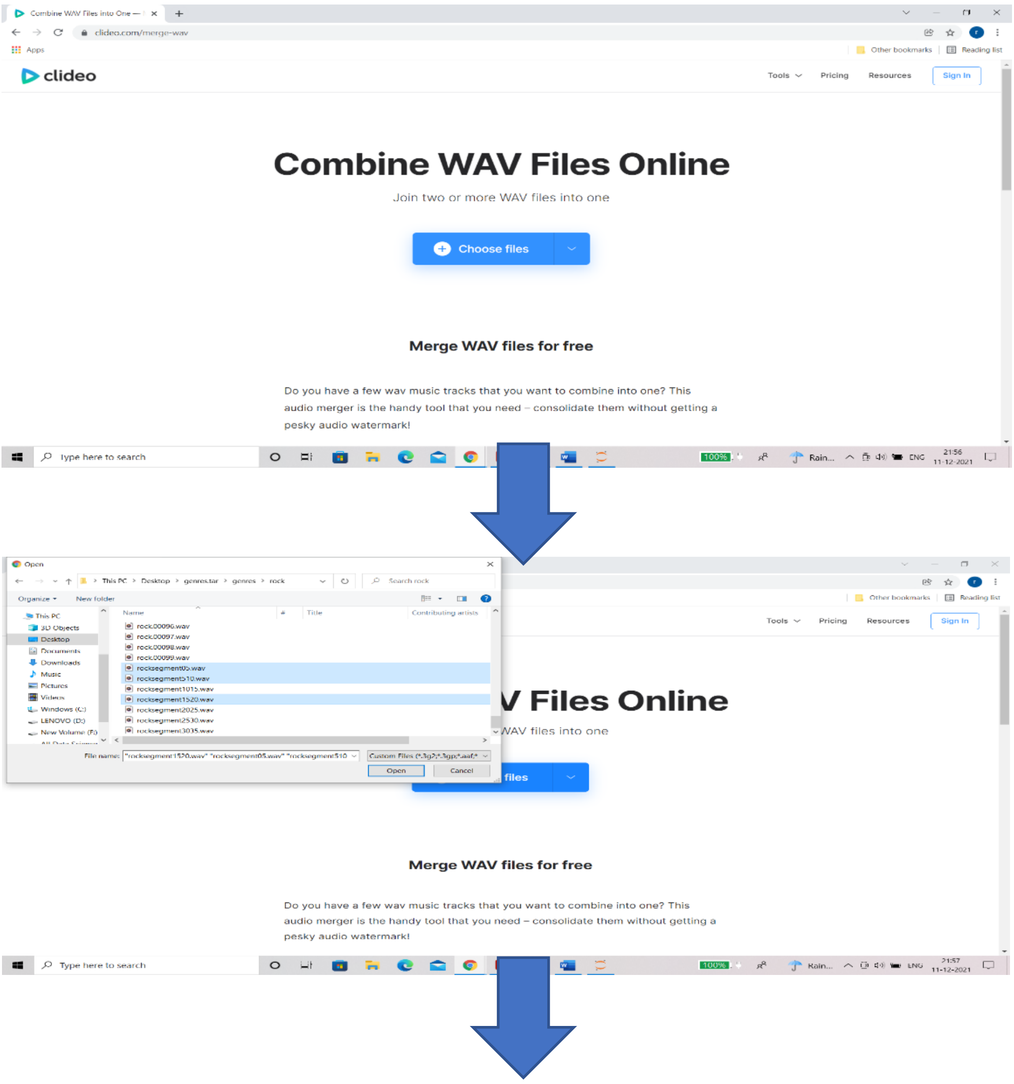
5.1 Data Pre-processing – The novelty in this project is in the pre-processing section. The libraries pydub, numpy, wavfile will need to be imported to run the pre-processing code.

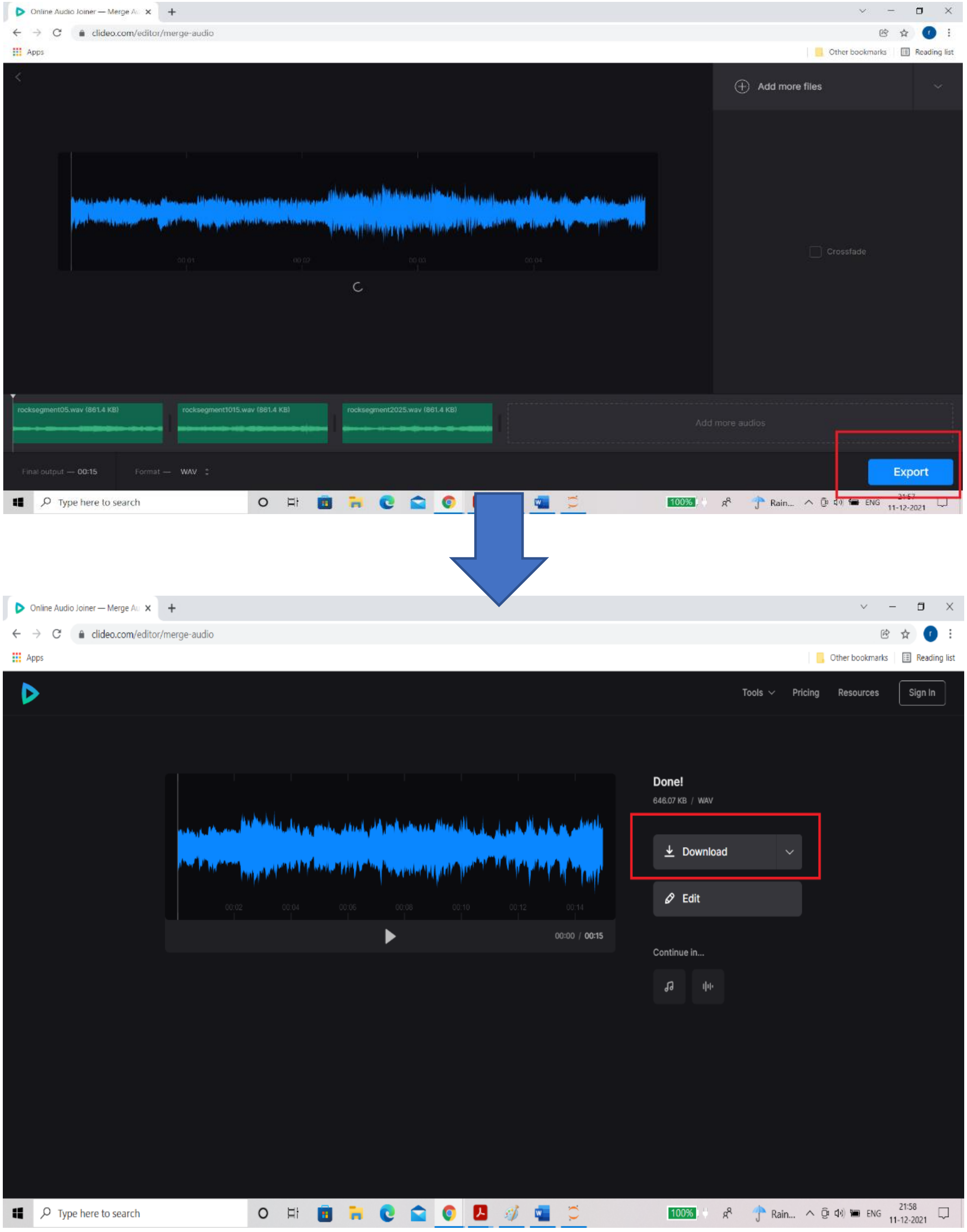
```
In [1]: from pydub import AudioSegment
        import numpy as np
        from scipy.io import wavfile
        from plotly.offline import init_notebook_mode
        import plotly.graph_objs as go
        import plotly

C:\Users\Rounak Sathe\anaconda3\lib\site-packages\pydub\utils.py:170: RuntimeWarning: Couldn't find ffmpeg or avconv - defaulti
ng to ffmpeg, but may not work
  warn("Couldn't find ffmpeg or avconv - defaulting to ffmpeg, but may not work", RuntimeWarning)
```

```
In [124]: fs, signal = wavfile.read("C:/Users/Rounak Sathe/Desktop/genres.tar/genres/rock/rock.00093.wav")
          signal = signal / (2**15)
          signal_len = len(signal)
          segment_size_t = 5 # segment size in seconds
          segment_size = segment_size_t * fs  # segment size in samples
          # Break signal into list of segments in a single-line Python code
          segments = np.array([signal[x:x + segment_size] for x in
                               np.arange(0, signal_len, segment_size)])
          # Save each segment in a seperate filename
          for iS, s in enumerate(segments):
              wavfile.write("C:/Users/Rounak Sathe/Desktop/genres.tar/genres/rock/rocksegment{0:d}{1:d}.wav".format(segment_size_t * iS,
                                                              segment_size_t * (iS + 1)), fs, (s))
```

The above code will first split the audio file into segments of 5 seconds each, following which the files will need to be merged. Audio segment is the library that can be used for the merging of the files. Since there was an error using this library, the files were merged manually using the clideo website.

5.2 Clideo website – The following screenshots show how audio files have been merged and downloaded.

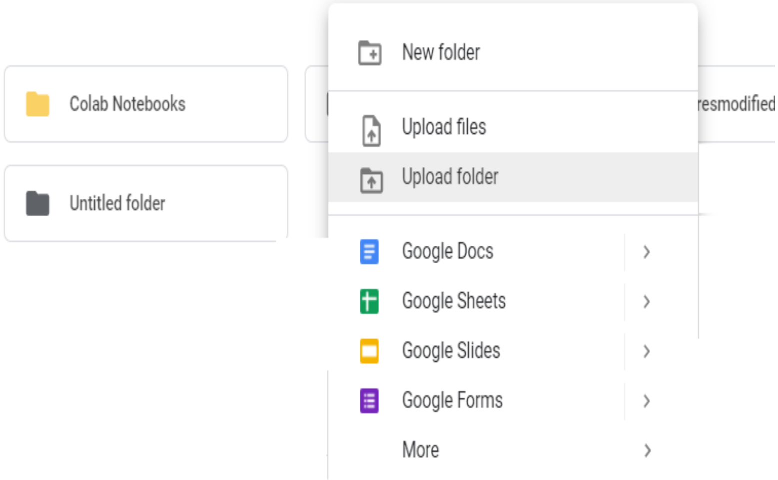




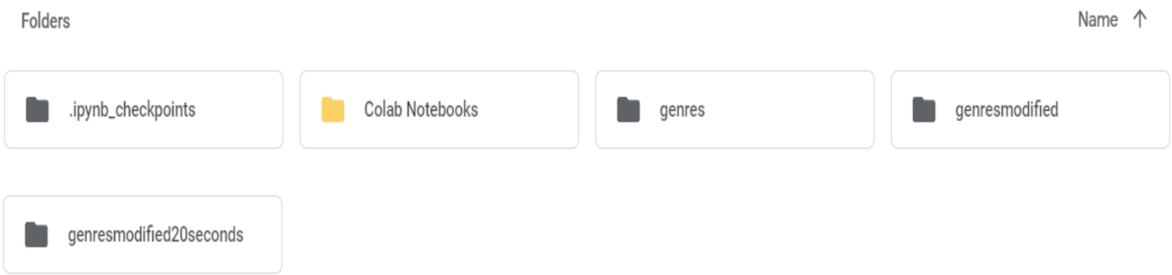
Following this step, the processed audio file can be downloaded. The new dataset will then need to be uploaded to drive and connected to the google colab just like before.

5.4 Google Drive –

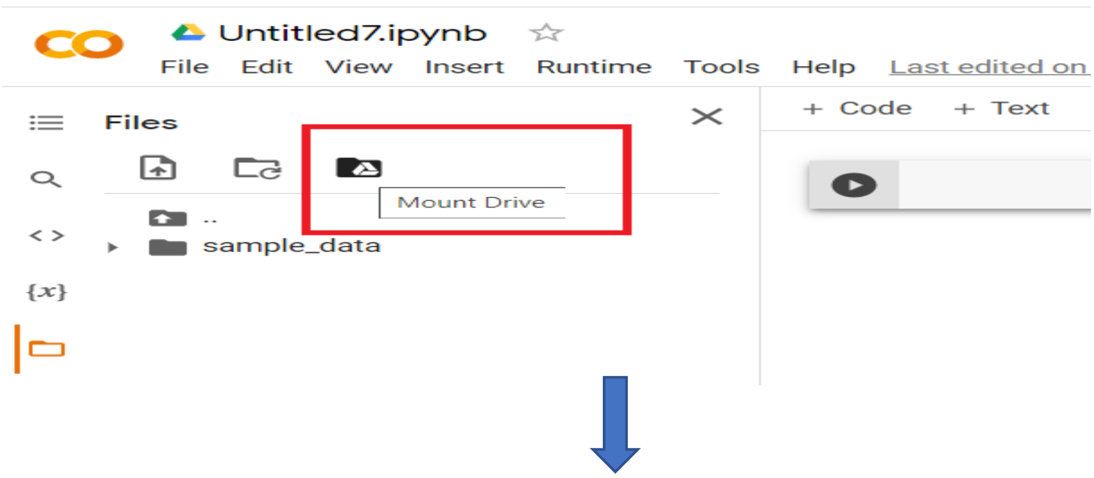
The above step will generate 1000 songs of 15 secs each and 1000 songs of 20 seconds each which will be placed in the folders genresmodified and genresmodified20seconds respectively along with the original dataset folder called genres. The following screenshots show how the dataset folders can be uploaded. After going to my drive section, press right click on the mouse to get this window –

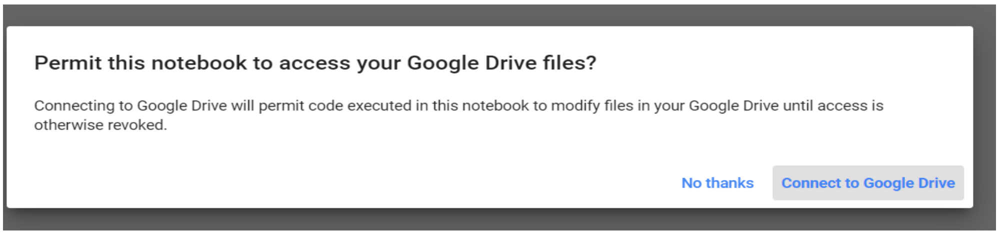


After following the above process for both the genres, genresmodified and genresmodified20seconds folders, there will be 3 folders present in my drive like this –



Google Colab – Google colab provides easy access to the google drive.





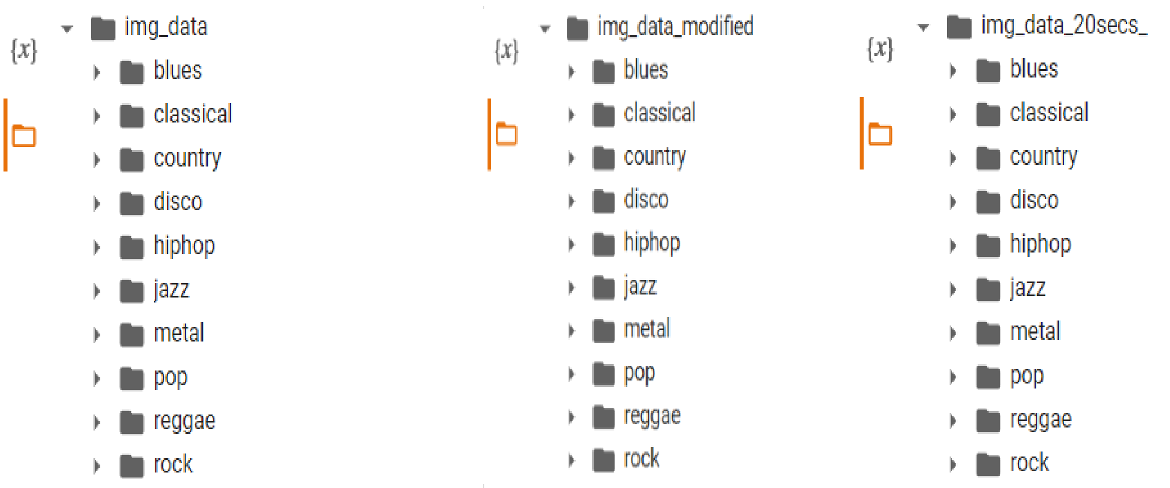
The above steps will connect the drive to the colab notebook following which the dataset can be accessed.

5.5 Colab Implementation –

There are 3 notebooks in total. The code in both the notebooks is the same except for the part where the dataset is loaded. The below diagram shows some of the libraries that will need to be imported. The next step is to convert the sound files into images.

```
import pandas as pd
import numpy as np
from numpy import argmax
import matplotlib.pyplot as plt
%matplotlib inline
import librosa
import librosa.display
import IPython.display
import random
import warnings
import os
from PIL import Image
import pathlib
import csv
# sklearn Preprocessing
from sklearn.model_selection import train_test_split
#Keras
import keras
import warnings
import tensorflow
warnings.filterwarnings('ignore')
from keras import layers
from keras.layers import Activation, Dense, Dropout, Conv2D, Flatten, MaxPooling2D, GlobalMaxPooling2D, GlobalAveragePooling1D, AveragePooling2D, Input, Add
from keras.models import Sequential
from tensorflow.keras.optimizers import SGD
```

- The first step is to create img_data, img_data_modified and img_data_20secs_ folders to store the spectrogram images. The directories will look like this –



The following code screenshot shows the code that will generate the above folders. As can be seen, the only difference is highlighted using a red square.

```
genres = 'blues classical country disco hiphop jazz metal pop reggae rock'.split()
for g in genres:
  pathlib.Path(f'img_data_20secs_/{g}').mkdir(parents=True, exist_ok=True)
  for filename in os.listdir(f'./drive/My Drive/genresmodified20seconds/{g}'):
        songname = f'./drive/My Drive/genresmodified20seconds/{g}/{filename}'
        y, sr = librosa.load(songname, mono=True, duration=5)
        print(y.shape)
        plt.specgram(y, NFFT=2048, Fs=2, Fc=0, noverlap=128, sides='default', mode='default', scale='dB');
        plt.axis('off');
        plt.savefig(f'img_data_20secs_/{g}/{filename[:-3].replace(".", "")}.png')
        plt.clf()
```

```
[2] genres = 'blues classical country disco hiphop jazz metal pop reggae rock'.split()
    for g in genres:
      pathlib.Path(f'img_data/{g}').mkdir(parents=True, exist_ok=True)
      for filename in os.listdir(f'./drive/My Drive/genres/{g}'):
            songname = f'./drive/My Drive/genres/{g}/{filename}'
            y, sr = librosa.load(songname, mono=True, duration=5)
            print(y.shape)
            plt.specgram(y, NFFT=2048, Fs=2, Fc=0, noverlap=128, sides='default', mode='default', scale='dB');
            plt.axis('off');
            plt.savefig(f'img_data/{g}/{filename[:-3].replace(".", "")}.png')
            plt.clf()
```

```
genres = 'blues classical country disco hiphop jazz metal pop reggae rock'.split()
for g in genres:
  pathlib.Path(f'img_data_modified/{g}').mkdir(parents=True, exist_ok=True)
  for filename in os.listdir(f'./drive/My Drive/genresmodified/{g}'):
        songname = f'./drive/My Drive/genresmodified/{g}/{filename}'
        y, sr = librosa.load(songname, mono=True, duration=5)
        print(y.shape)
        plt.specgram(y, NFFT=2048, Fs=2, Fc=0, noverlap=128, sides='default', mode='default', scale='dB');
        plt.axis('off');
        plt.savefig(f'img_data_modified/{g}/{filename[:-3].replace(".", "")}.png')
        plt.clf()
```

Following this, the next step is to split the folders in train and test folders. The following screenshot shows how to split the folders.

```
[ ] import splitfolders
    # To only split into training and validation set, set a tuple to `ratio`, i.e, `(.8, .2)`.
    splitfolders.ratio('./img_data_modified/', output="./datamodified", seed=1337, ratio=(.8, .2)) # default values

    Copying files: 979 files [00:00, 3251.60 files/s]

[ ] pip install split_folders

    Requirement already satisfied: split_folders in /usr/local/lib/python3.7/dist-packages (0.4.3)
```

The line pip install split_folders must be implemented first in case it is not already installed.

This stage is followed by image augmentation, where the ImageDataGenerator must be used for this process.

```python
from keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(
        rescale=1./255, #rescaled
        shear_range=0.2, #random transformations
        zoom_range=0.2, #zoom
        horizontal_flip=True) # horizontal flip
test_datagen = ImageDataGenerator(rescale=1./255)
```

- **Modelling stage –**

In this stage, the model building has been explained. First the libraries will need to be imported as shown below –

```python
from keras.layers import Input, Lambda, Dense, Flatten
from keras.models import Model
from keras.applications.vgg16 import VGG16
from keras.applications.vgg16 import preprocess_input
from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
import numpy as np
from glob import glob
import matplotlib.pyplot as plt

# re-size all the images to this
IMAGE_SIZE = [64, 64]
```

After this, the VGG16 model architecture is defined, followed by which the model will be built.

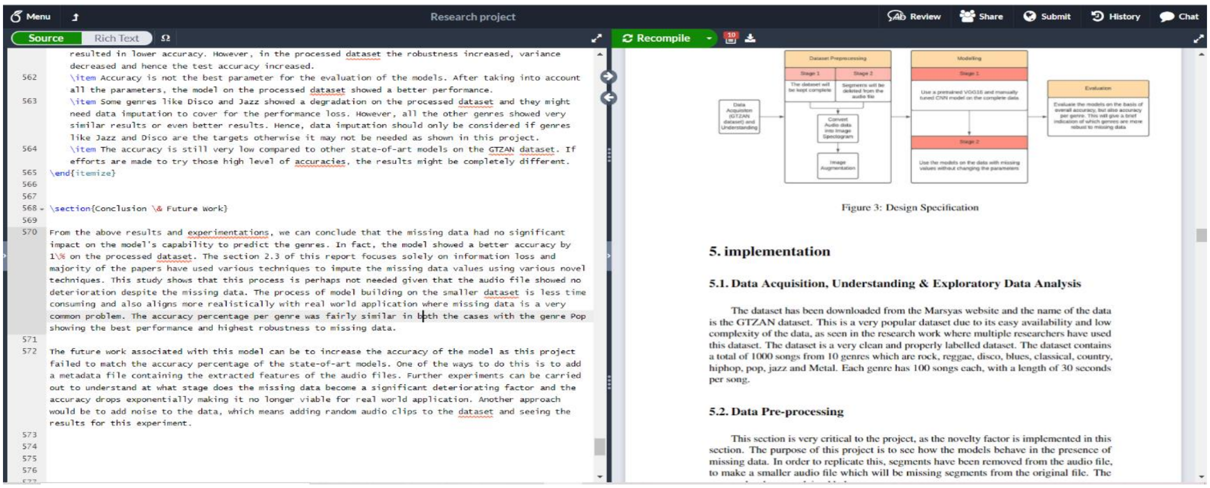
```python
[ ] IMAGE_SIZE = [64, 64]

[ ] vgg = VGG16(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)

    # don't train existing weights
    for layer in vgg.layers:
      layer.trainable = False

[ ] folders = glob('./datamodified/train/*')

[ ] # our layers - you can add more if you want
    x = Flatten()(vgg.output)
    x = Dense(1000, activation='relu')(x)
    prediction = Dense(len(folders), activation='softmax')(x)

[ ] # create a model object
    model = Model(inputs=vgg.input, outputs=prediction)

    # view the structure of the model
    model.summary()

    # tell the model what cost and optimization method to use
    model.compile(
      loss='categorical_crossentropy',
      optimizer='adam',
      metrics=['accuracy']
    )
```

```
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 64, 64, 3)]       0

block1_conv1 (Conv2D)        (None, 64, 64, 64)        1792

block1_conv2 (Conv2D)        (None, 64, 64, 64)        36928

block1_pool (MaxPooling2D)   (None, 32, 32, 64)        0

block2_conv1 (Conv2D)        (None, 32, 32, 128)       73856

block2_conv2 (Conv2D)        (None, 32, 32, 128)       147584

block2_pool (MaxPooling2D)   (None, 16, 16, 128)       0

block3_conv1 (Conv2D)        (None, 16, 16, 256)       295168

block3_conv2 (Conv2D)        (None, 16, 16, 256)       590080

block3_conv3 (Conv2D)        (None, 16, 16, 256)       590080

block3_pool (MaxPooling2D)   (None, 8, 8, 256)         0

block4_conv1 (Conv2D)        (None, 8, 8, 512)         1180160

block4_conv2 (Conv2D)        (None, 8, 8, 512)         2359808

block4_conv3 (Conv2D)        (None, 8, 8, 512)         2359808

block4_pool (MaxPooling2D)   (None, 4, 4, 512)         0

block5_conv1 (Conv2D)        (None, 4, 4, 512)         2359808

block5_conv2 (Conv2D)        (None, 4, 4, 512)         2359808

block5_conv3 (Conv2D)        (None, 4, 4, 512)         2359808

block5_pool (MaxPooling2D)   (None, 2, 2, 512)         0

flatten_1 (Flatten)          (None, 2048)              0

dense_2 (Dense)              (None, 1000)              2049000

dense_3 (Dense)              (None, 10)                10010

=================================================================
Total params: 16,773,698
Trainable params: 2,059,010
Non-trainable params: 14,714,688
_________________________________________________________________
```

After running the model for 1000 epochs, the outputs were generated along with a csv file that contains all the predictions from the model.

6. Overleaf –

This is cloud-based tool that has been used to write the project report. The following screenshot shows how the tool looks –



7. References –

https://keras.io/api/applications/vgg/

https://clideo.com/merge-audio

https://www.overleaf.com/