A Project Report

on

MUSIC GENRE CLASSIFICATION USING MACHINE LEARNING ALGORITHMS

Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

by

17WH1A0597	**Ms. B.CHARITHA**
17WH1A0586	**Ms. G.ANITHA**
18WH5A0514	**Ms. K.ANJALI**

under the esteemed guidance of

Ms. G.SHANTI

Assistant Professor

Department of Computer Science and Engineering

BVRIT HYDERABAD

College of Engineering for Women

(NBA Accredited – EEE, ECE, CSE and IT)

(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)

Bachupally, Hyderabad – 500090

June, 2021

DECLARATION

We hereby declare that the work presented in this project entitled **"MUSIC GENRE CLASSIFICATION USING MACHINE LEARNING ALGORITMS"** submitted towards completion of Project Work in IV year of B.Tech., CSE at 'BVRIT HYDERABAD College of Engineering For Women', Hyderabad is an authentic record of our original work carried out under the guidance of Ms. G. Shanti, Assistant Professor, Department of CSE.

Sign. with date:

Ms. B.CHARITHA

(17WH1A0597)

Sign. with date:

Ms. G.ANITHA

(17WH1A0586)

Sign. with date:

Ms. K.ANJALI

(18WH5A0514)

BVRIT HYDERABAD

College of Engineering for Women

(NBA Accredited – EEE, ECE, CSE and IT)

(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)

Bachupally, Hyderabad – 500090

Department of Computer Science and Engineering

Certificate

This is to certify that the Project Work report on "**MUSIC GENRE CLASSIFICATION USING MACHINE LEARNING ALGORITHMS**" is a bonafide work carried out by Ms. B.CHARITHA (17WH1A0597); Ms. G.ANITHA (17WH1A0586); Ms. K.ANJALI (18WH5A0514) in the partial fulfillment for the award of B.Tech. degree in **Computer Science and Engineering, BVRIT HYDERABAD College of Engineering for Women, Bachupally, Hyderabad**, affiliated to Jawaharlal Nehru Technological University Hyderabad, Hyderabad under my guidance and supervision.

The results embodied in the project work have not been submitted to any other University or Institute for the award of any degree or diploma.

Head of the Department	**Guide**
Dr. K. Srinivasa Reddy	**Ms. G.Shanti**
Professor and HoD	**Assistant Professor**
Department of CSE	

External Examiner

Acknowledgements

We would like to express our sincere thanks to **Dr. K V N Sunitha, Principal**, **BVRIT HYDERABAD College of Engineering for Women**, for providing the working facilities in the college.

Our sincere thanks and gratitude to our **Dr. K.Srinivasa Reddy, Professor and HoD**, Department of CSE, **BVRIT HYDERABAD College of Engineering for Women** for all the timely support and valuable suggestions during the period of our project.

We are extremely thankful and indebted to our internal guide, **Ms. G.Shanti, Assistant Professor**, Department of CSE**, BVRIT HYDERABAD College of Engineering for Women** for her constant guidance, encouragement and moral support throughout the project.

Finally, we would also like to thank our Project Coordinator, all the faculty and staff of **CSE** Department who helped us directly or indirectly, parents and friends for their cooperation in completing the project work.

Ms. B.CHARITHA
(17WH1A0597)

Ms. G.ANITHA
(17WH1A0586)

Ms. K.ANJALI
(18WH5A0514)

ABSTRACT

Music Genre Classification aims to classify the audio files to a particular category, for example, blues, country, classical, hip-hop, rock, disco, etc. Classifying music files is a challenging task and cannot be done manually. This project automates the classification of audio files by extracting spectral features such as spectral centroid, spectral contrast, spectral rolloff, spectral bandwidth, etc, classifying the audio files based on the extracted features. Classification methods like K Nearest Neighbours, Naïve Bayes, Decision Tree, Random Forest, Support Vector Machines were employed. Furthermore, Convolutional Neural Network – CNN is also used to classify music genres, where spectrograms of audio files are generated. These spectrograms are used as input images to the CNN. By training the CNN, features are extracted from the spectrograms and a fully connected layer of genres is generated to classify music genres.

LIST OF FIGURES

TABLE OF CONTENTS

INTRODUCTION

Music Genre Classification is the process of categorizing the music clips into different music genres.

1.1 Objective

The system allows the user to upload a music clip, classifies the music genre. This is done using Convolutional neural networks and classification algorithms like K Nearest Neighbours, Support Vector Machines, Random Forest, Naïve Bayes, Decision Tree.

1.2 Dataset

The dataset used is GTZAN, which is a collection of 1000 tracks of 10 different genres which has approximately 100 tracks per genre. There are 10 classes (10 music genres) each containing 100 audio tracks .Each track is in .wav format. It contains audio files of the following 10 genres:

- Blues
- Classical
- Country
- Disco
- Hip-hop
- Jazz
- Metal
- Pop
- Reggae
- Rock

Reason for choosing this dataset is that, this is most standard dataset used for music classification. One of the crucial step in the whole process is to extract the right features that can help us in distinguishing all these genres with the best prediction rate.

source :- http://marsyas.info/downloads/datasets.html

2. THEORETICAL ANALYSIS OF THE PROPOSED PROJECT

2.1 Requirements Gathering

2.1.1 Software Requirements

Programming Language : Python 3.6

Dataset : GTZAN Genre Collection

Packages : Librosa, Numpy, Pandas, Matplotlib, Scikit-learn

Tool : Colab Notebook

2.1.2 Hardware Requirements

Operating System: Windows 10

Processor : Intel Core i3-2348M

Memory : 4 GB (RAM)

2.2 Technologies Description

Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

Librosa

Librosa is a python package for music and audio analysis. It provides the building blocks necessary to create music information retrieval system.

- librosa.display

  visualization and display routines using matplotlib.

- librosa.feature

  Feature extraction and manipulation. This includes low-level feature extraction, such as chroma grams, Mel spectrogram, MFCC, and various other spectral and rhythmic features.

Numpy

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

Scikit – learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. The library is built upon the SciPy (Scientific Python) that must be installed before using scikit-learn.

.

3. FEATURE EXTRACTION

Feature extraction is the most crucial task in classification. The accuracy of correctly classifying a data mainly depends on the data that we are using and features extracted from that data to classify each instance. In the music signal processing, feature extraction methods can be classified into several aspects. Digital signal processing – on the time domain and frequency domain is one of these. Another useful strategy used for feature extraction is statistical descriptors like mean, median, standard deviation etc.

3.1 Zero Crossing Rate :

The Zero-Crossing Rate (ZCR) of an audio frame is the rate of sign-changes of the signal during the frame. In other words, it is the number of times the signal changes value, from positive to negative and vice versa, divided by the length of the frame. This feature has been used heavily in both speech recognition and music information retrieval. It usually has higher values for highly percussive sounds like those in metal and rock.

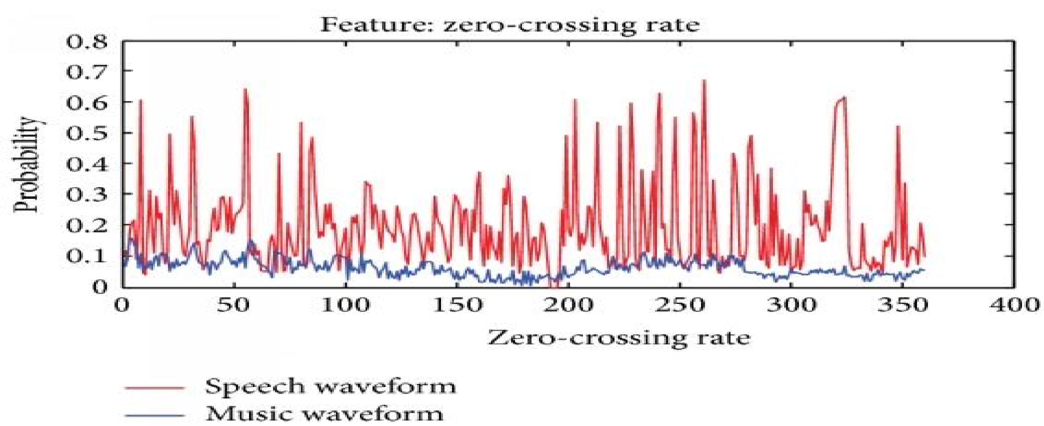


Fig 3.1.1 Zero-crossing rate

3.2 Spectral Centroid :

The spectral centroid and the spectral spread are two simple measures of spectral position and shape.
The spectral centroid is the center of 'gravity' of the spectrum.

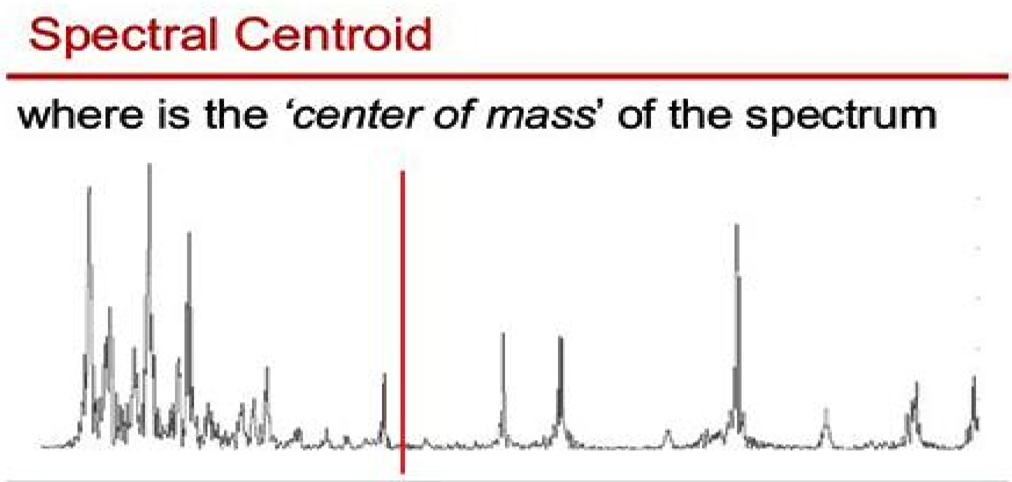


Fig 3.2.1: Spectral Centroid

Spectral spread measures how the spectrum is around its centroid. Obviously, low values of the spectral spread correspond to signals whose spectrum is tightly concentrated around the spectral centroid.

3.3 Spectral Contrast :

Spectral contrast is calculated as the difference between maximum and minimum magnitudes in the spectrum. It represents the decibel difference between the peak and the pit points on the spectrum of a signal. In audio processing, it gives information about the power changes in the sound.

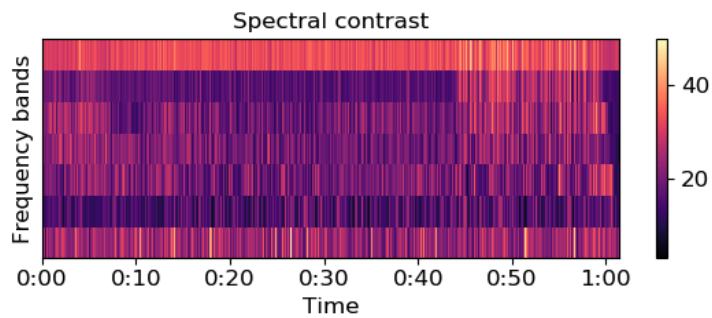


Fig 3.3.1 :Spectral Contrast

3.4 Spectral Bandwidth

Spectral bandwidth gives weighted average amplitude difference between frequency magnitude and brightness. It is an indication of the frequency range in the frame.

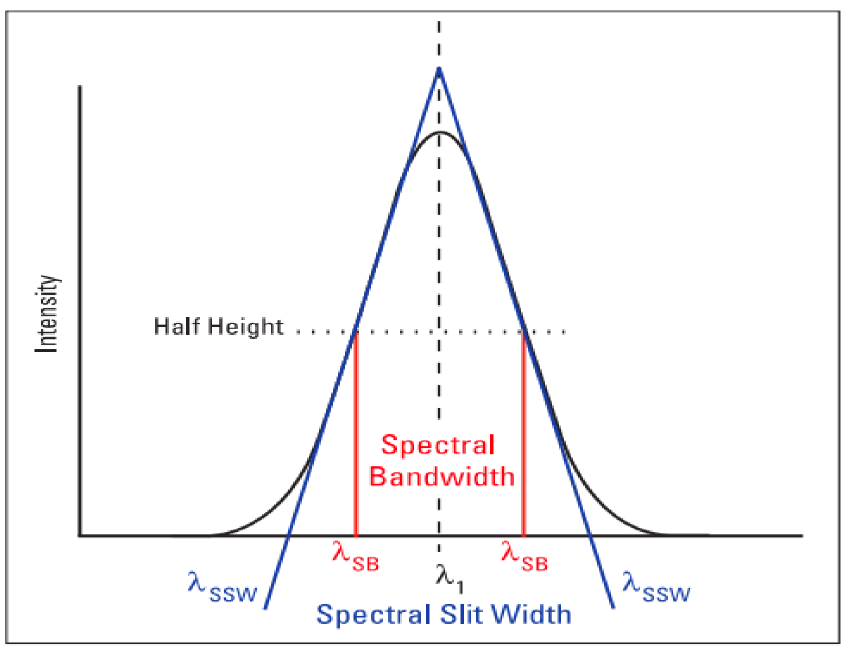


Figure 1: Gaussian intensity distribution of wavelengths emerging from the monochromator. The spectral bandwidth is defined by the red boundaries and λ_{SB}. The spectral slit width is depicted by the blue boundaries and λ_{SSW}.

Fig 3.4.1 Spectral Bandwidth

3.5 Spectral Roll-off :

Spectral roll-off is the normalized frequency at which the sum of the low frequency power values of the sound reaches a certain rate in the total power spectrum. Briefly, it can be defined as the frequency value corresponding to a certain ratio of the distribution in the spectrum. This rate is generally 85%.

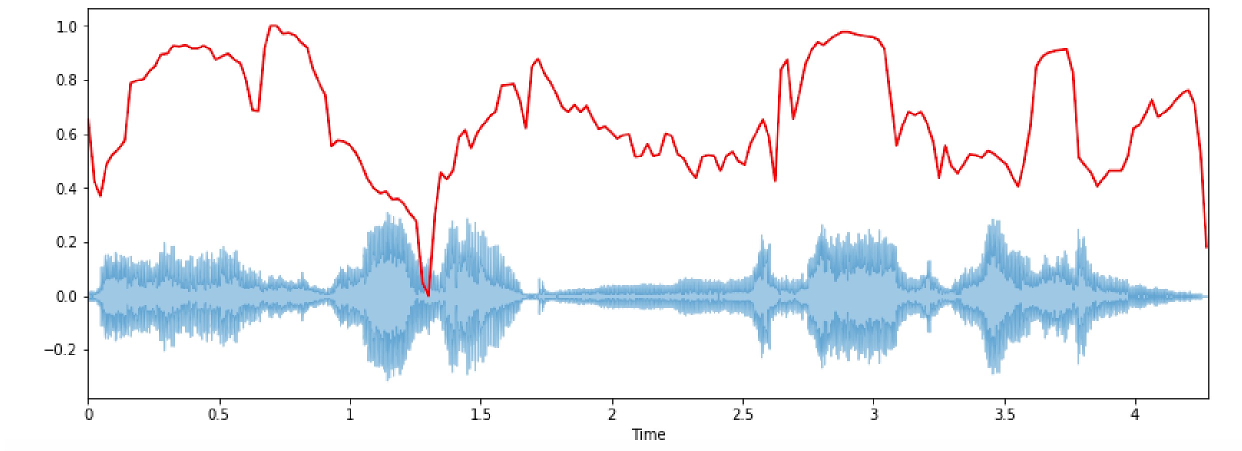


Fig 3.5.1 Spectral roll-off

3.6 Mel Frequency Coefficient of Cepstrum-MFCC :

MFCC represents a set of short term power spectrum characteristics of the sound and have been used in the state-of the-art recognition and sound categorization techniques. It models the characteristics of human voice. This features is a large part of the final feature vector (13 coefficients). The method to implement this feature is below :

– Dividing the signal into several short frames. The aim of this step is to keep an audio signal constant.

– For each frame, we calculated the periodogram estimate of the power spectrum. This is to know frequencies present in the short frames.

– Pushing the power spectra into the mel - filterbank and collecting the energy in each filter to sum it. We will then know the number of energy existing in the various frequency regions.

$$M(f) = 1125 \ln(1 + f/700)$$

i) Formula to work with MelScale

– Calculating the logarithm of the filterbank energies in the previous It enables humans to have our features closer to what humans can hear.

– Calculating the Discrete Cosine Transform (DCT) of the result. It decorrelates the filterbank energies with each others – Keep first 13 DCT coefficients. We remove the higher DCT coefficients which can introduce errors by representing changing in the filterbank energies.

4. METHODS

4.1 Machine Learning classifiers

To classify music according to their genre some machine learning algorithms are used. The classification algorithms used in this part are KNearest Neighbors, Naive Bayes, Decision Tree and Support Vector Machine, Random Forest.

4.1.1 K- Nearest Neighbors-KNN:

KNN is one of the distance based supervised learning algorithms. When solving the classification problem with this method, a model is not created and the test operation is performed on the labeled samples in the data set. A new instance of the class label will be calculated from the distance from the instances in the dataset. From these calculated distances, the class tag is estimated by voting on the class labels of the nearest k. When calculating the distance, the Manhattan distance formulas is used.

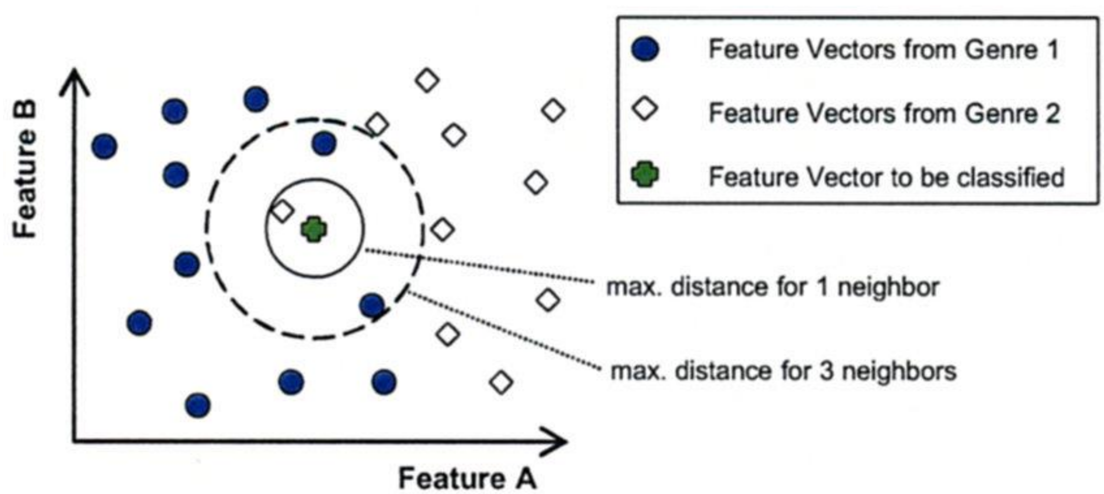


Fig 4.1.1 K- Nearest Neighbors Classifier(KNN)

4.1.2 Naive Bayes-NB:

It is a kind of classifier that works on Bayes theorem. Prediction of membership probabilities is made for every class such as the probability of data points associated to a particular class.The class having maximum probability is appraised as the most suitable class. This is also referred as Maximum A Posteriori (MAP).

The MAP for a hypothesis is:

$MAP\ (H)$ = max $((H|E))$

$MAP\ (H)$ = max $((H|E)\ *(P(H))\ /P(E))$

$MAP\ (H)$ = max$(P(E|H)\ *P(H))$

(E) is evidence probability, and it is used to normalize the result. The result will not be affected by removing (E). NB classifiers conclude that all the variables or features are not related to each other. The Existence or absence of a variable does not impact the existence or absence of any other variable. It is an algorithm that can be used in various problems because it is compatible with every kind of data and simple statistical calculations are required.

4.1.3 Decision Tree-DT:

Decision trees are learning algorithms that provide a supervised and model based approach. It tries to identify the most distinctive feature in the data set as the root node of the tree. An entropy calculation is made when the most distinguishing feature is found. There are also different metrics in the literature that provide differentiating features.

4.1.4 Support Vector Machine-SVM:

SVM is one of model based supervised learning algorithms. DVM is based on the principle of training for a decision surface that will allow the two classes to distinguish one another. This decision surface is created by optimizing the boundary regions of the two classes. SVM can be used in multi-class data sets other than two-class data sets.

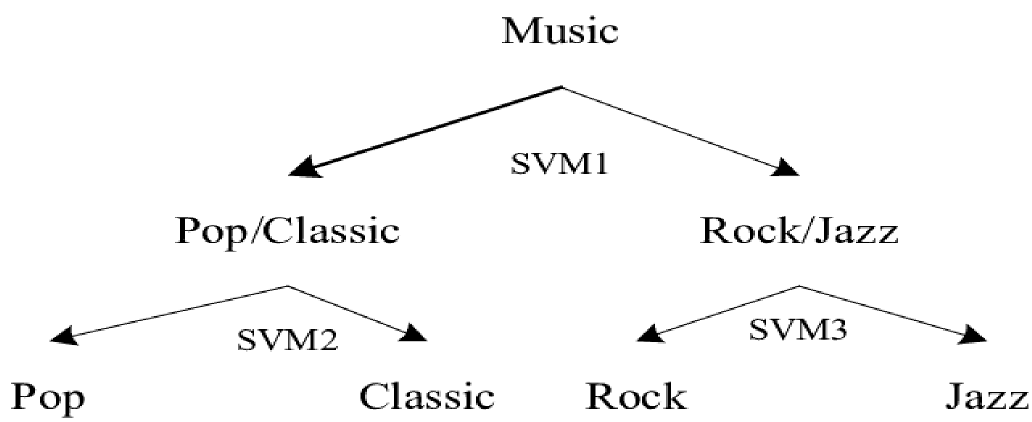


Fig 4.4.1 Classification of Music Genre using SVM

4.1.5 Random Forest-RF:

Random Forest (RF) is also utilized to the same feature set to search the success of an ensemble technique as to the music genre classification. RF can be used as a combination of multiple decision trees with bagging sample selection strategy.″Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset.″ Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of over-fitting.

4.2. Convolutional Neural Network

Convolutional neural networks is a tool, used to classify items that contain spatial neighborhood. Array of randomly created filters are used in the process and they are tweaked to better describe the data. Normally they are used to classify images but one dimensional filters can be utilized to classify audio. Also two dimensional filters can be used in the CNNs with spectrograms of audio. Several CNNs with general configuration given are trained and fine-tuned to classify a set of songs. Numbers of convolutional layers, fully connected layers, filters in convolutional layers changes between these networks. Output of a 10 fully connected layer is saved for each song. This output is used as feature vector in similarity calculations.

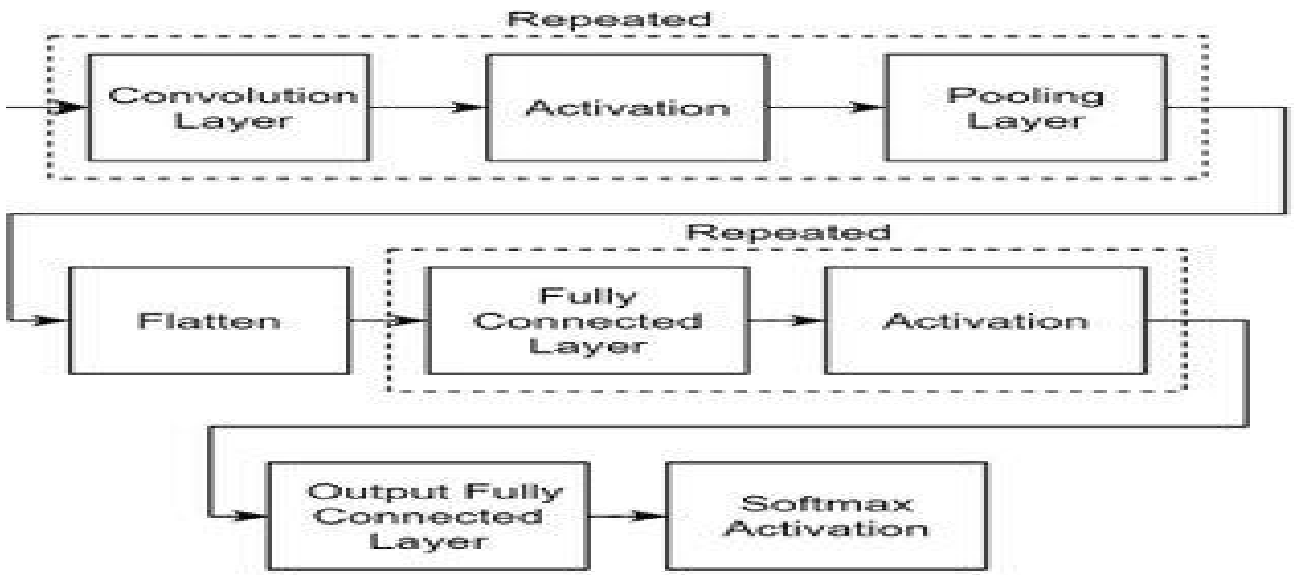


Fig 4.2.1 Convolutional Neural Network Configuration

After fine tuning, it is decided on a network with following configuration:

1- 2D Convolution Layer, 5x5 sized 32 filters, LeakyReLU activation function

2- Max Pooling Layer

3- 2D Convolution Layer, 5x5 sized 32 filters, LeakyReLU activation function

4- Max Pooling Layer

5- 2D Convolution Layer, 5x5 sized 32 filters, LeakyReLU activation function

6- Max Pooling Layer

7- 2D Convolution Layer, 5x5 sized 32 filters, LeakyReLU activation function

8- Average Pooling Layer

9- Flatten Layer

10- Dense Layer, 256 nodes, LeakyReLU activation function

11- Dense Layer, 128 nodes, LeakyReLU activation function

12- Dense Layer, 64 nodes, LeakyReLU activation function

13- Dense Layer, 10 nodes, Softmax activation function, as output layer

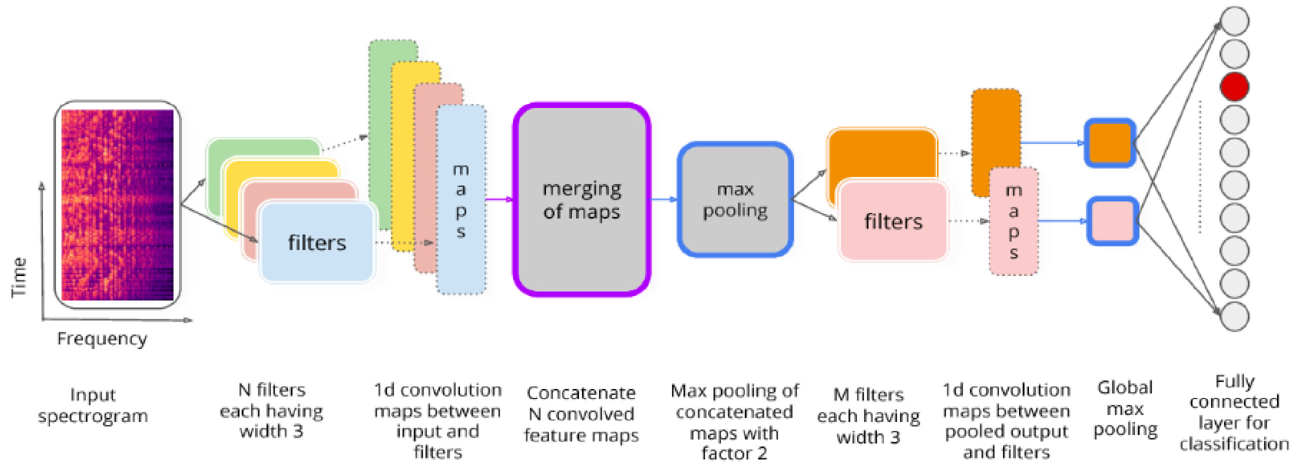


Fig 4.2.2 Convolutional Neural Network for Music Genre Classification

Raw audio data and acoustic features extracted from the audio is given to the network as input. Configuration of this features are given below.

CNN architectures vary with the type of the problem at hand. The proposed model consists of three convolutional layers each followed by a maxpooling layer. The final layer is fully connected MLP. LeakyReLu activation function is applied to the output of every convolutional layer and fully connected layer. The first convolutional layer filters the input image with 32 kernels of size 3x3. After max pooling is applied, the output is given as an input for the second convolutional layer with 64 kernels of size 4x4. The last convolutional layer has 128 kernels of size 1x1 followed by a fully connected layer of 512 neurons. The output of this layer is given to softmax function which produces a probability distribution of the four output class. The model is trained using adaptive moment estimation (Adam) with batch size of 100 for 1000 epochs.

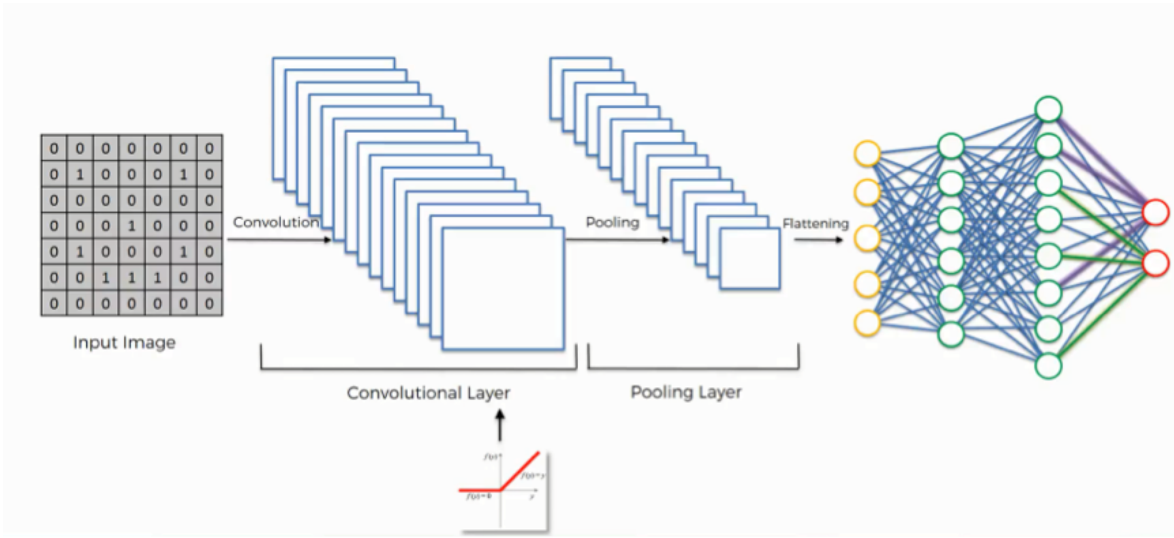


Fig 4.2.3: Convolution Neural Network

There are four CNN algorithm steps,

Convolution:

The term convolution refers to the mathematical combination of two functions toproduce a third function. It merges two sets of information. In the case of a CNN, the convolutionis performed on the input data with the use of a filter or kernel to then produce a feature map.

Here are the three elements that enter into the convolution operation:

- Input image
- Feature detector
- Feature map

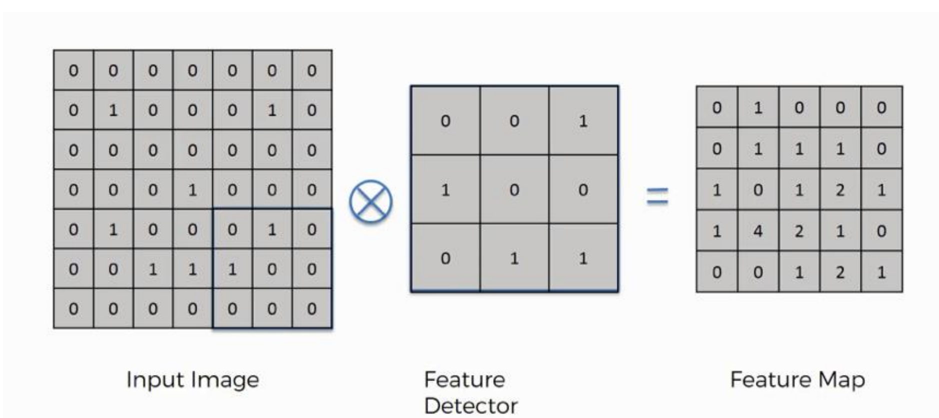


Fig 4.2.4: Convolution in CNN

Max pooling:

Max pooling is a sample-based discretization process. The objective is to down-sample an input representation(image, hidden-layer output matrix, etc.), reducing its dimensionality and allowing for assumptions to bemade about features contained in the sub-regions binned.

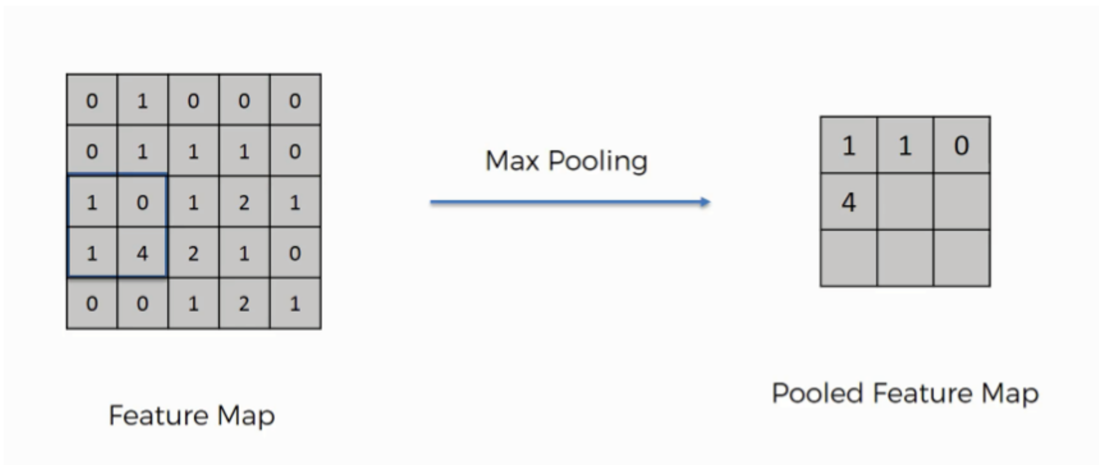


Fig 4.2.5: Max Pooling in CNN

Flattening:

Flattening is the process of converting all the resultant 2 dimensional arrays into asingle long continuous linear vector.

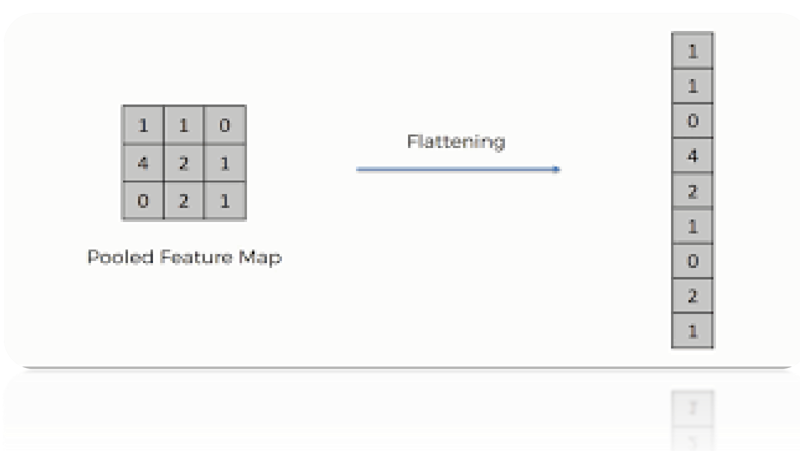


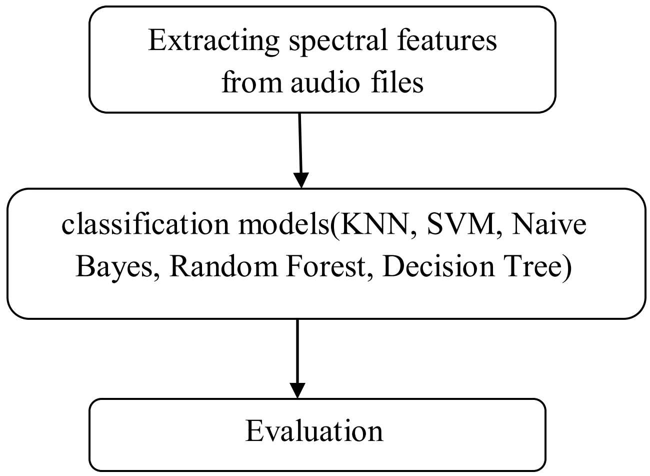
Fig 4.2.6: Flattening in CNN

Full Connection:

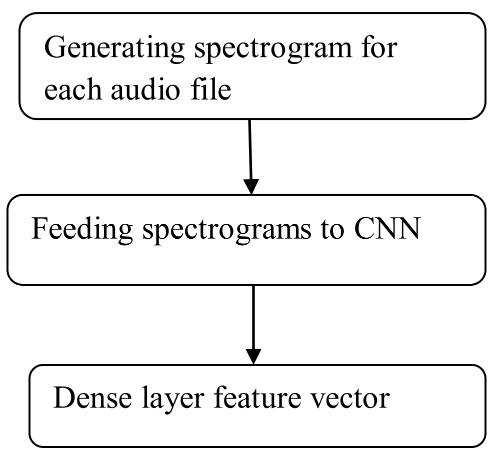
At the end of a CNN, the output of the last Pooling Layer acts as a input to this Fully Connected Layer. There can be one or more of these layers ("fully connected" means that every node in the first layer is connected to every node in the second layer).

5. DESIGN

5.1 Architecture Diagram



5.1 Architecture Diagram for CNN



6. IMPLEMENTATION

6.1 Coding

Music Genre Classification using Machine Learning algorithms

```
from google.colab import drive
drive.mount('/content/drive')

import librosa
import pandas as pd
import numpy as np
import csv
import os
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
import keras
from scipy import signal
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn import metrics
import IPython.display as ipd

genres = 'blues classical country disco hiphop jazz metal pop reggae rock'.split()

header = 'filename zero_crossing_rate_mean zero_crossing_rate_median zero_crossing_rate_std
spectral_centroid_mean spectral_centroid_median spectral_centroid_std spectral_contrast_mean
```

```
spectral_contrast_median spectral_contrast_std spectral_bandwidth_mean spectral_bandwidth_
median spectral_bandwidth_std spectral_rolloff_mean spectral_rolloff_median spectral_rolloff_s
td'
for i in range(1, 21):
    header += f' mfcc_mean{i}'
    header += f' mfcc_median{i}'
    header += f' mfcc_std{i}'
header += ' label'
header = header.split()

def feature_extraction() :
  featurefilename = 'dataset.csv'
  file = open(featurefilename, 'w', newline='')

  with file:
      writer = csv.writer(file)
      writer.writerow(header)
  genres = 'blues classical country disco hiphop jazz metal pop reggae rock'.split()
  for g in genres:
      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=30)
          zcr = librosa.feature.zero_crossing_rate(y)
          spec_cent = librosa.feature.spectral_centroid(y=y, sr=sr)
          spec_con = librosa.feature.spectral_contrast(y=y, sr=sr)
          spec_bw = librosa.feature.spectral_bandwidth(y=y, sr=sr)
          rolloff = librosa.feature.spectral_rolloff(y=y, sr=sr)

          mfcc = librosa.feature.mfcc(y=y, sr=sr)
```

```
        to_append = f'{filename} {np.mean(zcr)} {np.median(zcr)} {np.std(zcr)} {np.mean(spe
c_cent)} {np.median(spec_cent)} {np.std(spec_cent)} {np.mean(spec_con)} {np.median(spec_c
on)} {np.std(spec_con)} {np.mean(spec_bw)} {np.median(spec_bw)} {np.std(spec_bw)} {np.m
ean(rolloff)} {np.median(rolloff)} {np.std(rolloff)}'
      for e in mfcc:
        to_append += f' {np.mean(e)}'
        to_append += f' {np.median(e)}'
        to_append += f' {np.std(e)}'
      to_append += f' {g}'
      file = open('dataset.csv', 'a', newline='')
      with file:
        writer = csv.writer(file)
        writer.writerow(to_append.split())



def data_split() :
  data = pd.read_csv('dataset.csv')
  data.head()
  data = data.drop(['filename'],axis=1)
  genre_list = data.iloc[:, -1]
  encoder = LabelEncoder()
  y = encoder.fit_transform(genre_list)
  scaler = StandardScaler()
  X = scaler.fit_transform(np.array(data.iloc[:, :-1], dtype = float))
  x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
  return (x_train, x_test, y_train, y_test)


def KNN_model(x_train, x_test, y_train) :
  classifier= KNeighborsClassifier(n_neighbors= 3, p = 1)
```

```
  classifier.fit(x_train, y_train)
  pred = classifier.predict(x_test)
  return pred


def naive_bayes(x_train, x_test, y_train) :
  classifier = GaussianNB()
  classifier.fit(x_train, y_train)
  pred= classifier.predict(x_test)
  return pred

def decisionTree(x_train, x_test, y_train) :
  classifier= DecisionTreeClassifier(criterion='entropy', random_state=0)
  classifier.fit(x_train, y_train)
  pred= classifier.predict(x_test)
  return pred

def SVM_model(x_train, x_test, y_train ) :
  classifier = SVC(kernel="linear", random_state=0)
  classifier.fit(x_train, y_train)
  pred= classifier.predict(x_test)
  return pred

def accuracy(y_test, y_pred) :
  acc = accuracy_score(y_test, y_pred)
  return acc

def randomForest(x_train, x_test, y_train) :
  classifier= RandomForestClassifier()
  classifier.fit(x_train, y_train)
  pred= classifier.predict(x_test)
```

```
  return pred

result = {}

feature_extraction()

.(x_train, x_test, y_train, y_test) = data_split()

knn_pred = KNN_model(x_train, x_test, y_train)
print(knn_pred)
knn_acc = accuracy(y_test, knn_pred)
result['knn'] = knn_acc
print("\naccuracy = ",  knn_acc)
print("\nConfusion matrix")
print(metrics.confusion_matrix(y_test, knn_pred))


nb_pred = naive_bayes(x_train, x_test, y_train)
print(nb_pred)
nb_acc = accuracy(y_test, nb_pred)
result['nb'] = nb_acc
print("\naccuracy = ",  nb_acc)
print("\nConfusion matrix")
print(metrics.confusion_matrix(y_test, nb_pred))


dt_pred = decisionTree(x_train, x_test, y_train)
print(dt_pred)
dt_acc = accuracy(y_test, dt_pred)
result['dt'] = dt_acc
print("\naccuracy = ",  dt_acc)
```

```
print("\nConfusion matrix")
print(metrics.confusion_matrix(y_test,dt_pred))

rf_pred = randomForest(x_train, x_test, y_train)
print(rf_pred)
rf_acc = accuracy(y_test, rf_pred)
result['rf'] = rf_acc
print("\naccuracy = ",  rf_acc)
print("\nConfusion matrix")
print(metrics.confusion_matrix(y_test, rf_pred))


svm_pred = SVM_model(x_train, x_test, y_train)
from tabulate import tabulate
d = []
for i in range(200) :
  d.append([genres[y_test[i]], genres[svm_pred[i]]])

print(tabulate(d, headers=["Actual", "Predicted"]))

svm_acc = accuracy(y_test, svm_pred)
result['svm'] = svm_acc
print("\naccuracy = ",  svm_acc)
print("\nConfusion matrix")
print(metrics.confusion_matrix(y_test, svm_pred))

from tabulate import tabulate

d = [ ["K Nearesr Neighbours", result['knn']],
      ["Random Forest", result['rf']],
      ["Naive Bayes", result['nb']],
      ["Decision Tree", result['dt']],
```

```
    ["Support Vector Machine", result['svm']]]
print(tabulate(d, headers=["Classifier", "Accuracy"]))
```

Music Genre Classification using Convolutional Neural Networks

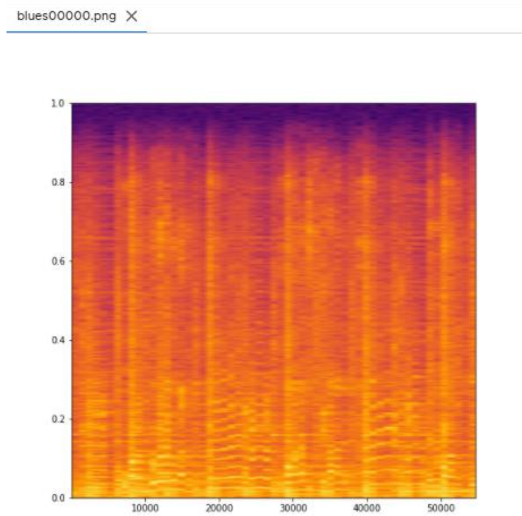
```
from google.colab import drive
drive.mount('/content/drive')

!pip install split-folders

import pandas as pd
import numpy as np
from numpy import argmax
import matplotlib.pyplot as plt
import librosa
import librosa.display
import IPython.display
import random
import warnings
import os
from PIL import Image
import pathlib
import csv
from sklearn.model_selection import train_test_split
import keras
from keras import layers
from keras.layers import Activation, Dense, Dropout, Conv2D, Flatten, MaxPooling2D, Global
MaxPooling2D, GlobalAveragePooling1D, AveragePooling2D, Input, Add
from keras.models import Sequential
from keras.optimizers import SGD
```

```
from keras.preprocessing.image import ImageDataGenerator
import IPython.display as ipd
import splitfolders

genres = 'blues classical country disco hiphop jazz metal pop reggae rock'.split()

cmap = plt.get_cmap('inferno')
plt.figure(figsize=(8,8))

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}'
        print(filename, songname)
        y, sr = librosa.load(songname, mono=True, duration=5)

        plt.specgram(y, NFFT=2048, Fs=2, Fc=0, noverlap=128, cmap=cmap, sides='default', mode
='default', scale='dB');
        plt.savefig(f'img_data/{g}/{filename[:-3].replace(".", "")}.png')

splitfolders.ratio('./img_data/', output="./data", ratio=(.8, .2))

train_datagen = ImageDataGenerator(
        rescale=1./255,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True)
test_datagen = ImageDataGenerator(rescale=1./255)

training_set = train_datagen.flow_from_directory(
        './data/train',
```

```
        target_size=(64, 64),
        batch_size=32,
        class_mode='categorical',
        shuffle = False)
test_set = test_datagen.flow_from_directory(  './data/val',
        target_size=(64, 64),
        batch_size=32,
        class_mode='categorical',
        shuffle = False )

model = Sequential()
input_shape=(64, 64, 3)
model.add(Conv2D(32, (3, 3), strides=(2, 2), input_shape=input_shape))
model.add(AveragePooling2D((2, 2), strides=(2,2)))
model.add(Activation('relu'))
model.add(Conv2D(64, (3, 3), padding="same"))
model.add(AveragePooling2D((2, 2), strides=(2,2)))
model.add(Activation('relu'))
model.add(Conv2D(64, (3, 3), padding="same"))
model.add(AveragePooling2D((2, 2), strides=(2,2)))
model.add(Activation('relu'))
model.add(Flatten())
model.add(Dropout(rate=0.5)).
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dropout(rate=0.5))
model.add(Dense(10))
model.add(Activation('softmax'))
model.summary()

epochs = 100
```

```
batch_size = 8
learning_rate = 0.01
decay_rate = learning_rate / epochs
momentum = 0.9
sgd = SGD(lr=learning_rate, momentum=momentum, decay=decay_rate)
model.compile(optimizer="sgd", loss="categorical_crossentropy", metrics=['accuracy'])
model.fit_generator(
    training_set,
    steps_per_epoch=10,
    epochs=100,
    validation_data=test_set,
    validation_steps=50)

model.evaluate_generator(generator=test_set, steps=100)
```

6.2 Training dataset screenshots



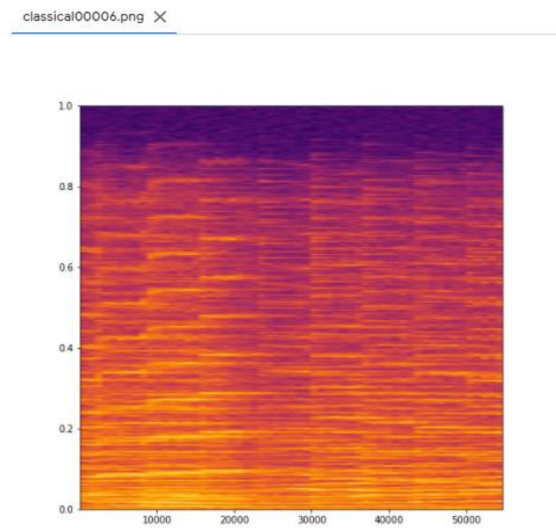
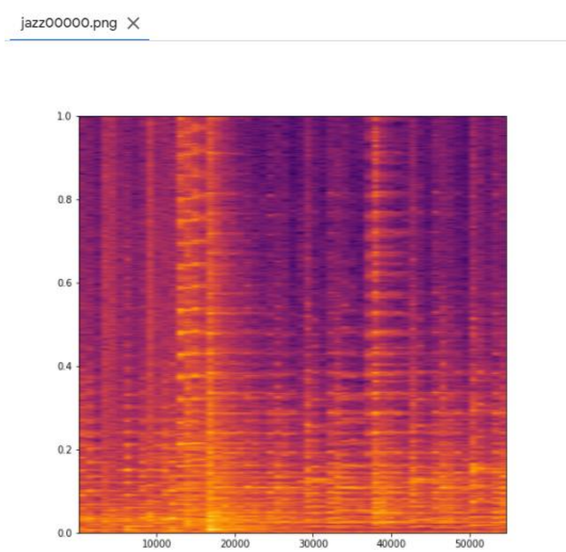


Fig 6.2.1 Spectrogram for blues genre

Fig 6.2.2Spectrogram for classical genre



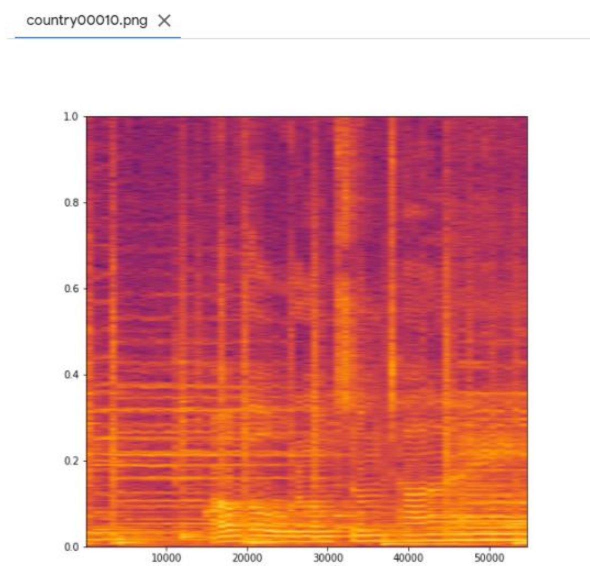


Fig 6.2.3 Spectrogram for jazz genre

Fig 6.2.4 Spectrogram for country genre

6.3 Output Screenshots

```
[37] from tabulate import tabulate

     d = [ ["K Nearesr Neighbours", result['knn']],
           ["Random Forest", result['rf']],
           ["Naive Bayes", result['nb']],
           ["Decision Tree", result['dt']],
           ["Support Vector Machine", result['svm']]]

     print(tabulate(d, headers=["Classifier", "Accuracy"]))

     Classifier                Accuracy
     ----------------------  ----------
     K Nearesr Neighbours         0.67
     Random Forest                0.65
     Naive Bayes                  0.45
     Decision Tree                0.465
     Support Vector Machine       0.69
```

Fig 6.3.1 Accuracy comparision

```
original, predicted = find_genre('/content/disco.00000.wav', 'svm')
print(original, genres[predicted])

disco disco
```

Fig 6.3.2 Output 1

```
original, predicted = find_genre('/content/disco.00000.wav', 'knn')
print(original, genres[predicted])

disco blues
```

Fig 6.3.3 Output 2

7. CONCLUSION AND FUTURE SCOPE

Conclusion

This project was aimed to tackle the problem of automatic music genre classification based on various features. Firstly extraction of spectral features from audio files is done by feature extraction and selection, lastly followed by classification. Here, we focussed our spectrum of features as these act as a good metric for human perception of music. Through feature analysis and classification, a maximum accuracy of 70% was obtained.

Future Scope

Further, we look forward to include more features into the application and improvise our classification algorithm to improve overall performance. We also plan to broaden the spectrum of the genres used.

8. APPLICATIONS

Apart from the most generic use of classifying huge chunks of data, this classifier can also be used for following applications

- Developing an automatic genre based disco lights system.
- .Automatic Equaliser.
- Emotion-mapped music player.
- recommender systems
- track separation
- instrument recognition,

9. REFERENCES

[1] Elbir, A. Bilal Cam, H, EmreIyican, M., Ozturk, B., &Aydin, N. (2018). Music Genre Classification and Recommendation by Using Machine Learning Techniques. 2018 Innovations in Intelligent Systems and Applications Conference.

[2] A. Tzanetakis, G. and Cook, P. "Musical genre classification of audio signal", IEEE Transactions on Speech and Audio Processing, Vol. 10, No. 3, pp. 293-302, July 2002.

[3] Karunakaran, N., &Arya, A. (2018). A Scalable Hybrid Classifier for Music Genre Classification using Machine Learning Concepts and Spark. 2018 International Conference on Intelligent Autonomous Systems

[4] Vishnupriya, S., &Meenakshi, K. (2018). Automatic Music Genre Classification using Convolution Neural Network. 2018 International Conference on Computer Communication and Informatics (ICCCI)

[5] Ahmet Elbir, Hamza Osman Ilhan, Gorkem Serbes, Nizamettin Aydin. "Short Time Fourier Transform based music Genre classification" , 2018 Electric Electronics, Computer Science, Biomedical Engineerings' Meeting (EBBT), 2018