A Minor Project Report

On

Music Genre Classification

Submitted in Partial Fulfilment of the requirement for the award of the degree of

BACHELOR OF TECHNOLGY

(Department of Computer Science and Engineering)

Submitted by

Akshat Kashyap (190970101009)

Under the Guidance of

……Mr. Manish Kumar ……..

(Assistant Professor)

in

Department of Computer Science and Engineering

THDC INSTITUTE OF HYDROPOWER ENGINEERING & TECHNOLOGY

TEHRI ,UTTRAKHAND,INDIA

(Uttarakhand Technical University, Dehradun)

2019-2023

TABLE OF CONTENTS

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled **"Music Genre Classification"** in partial fulfillment of the requirement for the award of degree of **Bachelor of Technology** and submitted in Department of **Computer Science and Engineering** of THDC Institute of Hydropower Engineering & Technology, Tehri,is an authentic record of my own work carried out .Under the supervision of Mr Manish kumar **,**Assistant Professor, Department of **Computer Science and Engineering**, THDC Institute of Hydropower Engineering & Technology, Tehri.

The matter presented in this report has not been submitted by me anywhere for the award of any other degree of this or any other institute.

Akshat Kashyap (190970101009)

This is to certify that the above statement made by the candidate is correct to the best of our knowledge.

Date:

(**Mr. Manish Kumar**)
Supervisor

ACKNOWLEDGEMENT

It gives me a great sense of pleasure to present the report of the B. Tech Project undertaken during B. Tech. Final Year. I owe special debt of gratitude to Professor **Mr. Manish Kumar,** Department of Computer Science and Engineering, THDC Institute of Hydropower Engineering & Technology ,Tehri for his constant support and guidance throughout the course of our work. His sincerity, thoroughness and perseverance have been a constant source of inspirationfor us. It is only his cognizant efforts that our endeavors have seen light of the day.We also take the opportunity to acknowledge the contribution of Professor **Mr. Manish Kumar** support and assistance during the development of the project.

I also do not like to miss the opportunity to acknowledge the contribution of all

I also do not like to miss the opportunity to acknowledge the contribution of all faculty members of the department for their kind assistance and cooperation during the development of our project. Last but not the least,i acknowledge our friends for their contribution in the completion of the project.

Akshat Kashyap

(190970101009)

ABSTRACT

About the project

Audio classification is an Application of machine learning where different sound is categorized in certain categories. our project problem statement as like given multiple audio files, and the task is to categorize each audio file in a certain category like audio belongs to Disco, hip-hop, etc.

The music genre classification can be built using different approaches in which the top 4 approaches that are mostly used are listed below.

1. Multiclass support vector machine
2. K-Nearest Neighbors
3. K-means clustering algorithm
4. Convolutional neural network

We will use K-Nearest Neighbors algorithm because various researches prove it is one of the best algorithms to give good performance and till time along with optimized models organizations uses this algorithm in recommendation systems as support.

K-Nearest Neighbor :- KNN is a machine learning algorithm used for regression, and classification. It is also known as the lazy learner algorithm. It simply uses a distance-based method o find the K number of similar neighbors to new data and the class in which the majority of neighbors lies, it results in that class as an output. Now let us get our system ready for project implementation.

Need of the project

Audio processing is one of the most complex tasks in data science as compared to image processing and other classification techniques. One such application is music genre classification which aims to classify the audio files in certain categories of sound to which they belong. The application is very important and requires automation to reduce the manual error and time because if we have to classify the music manually then one has to listen out each file for the complete duration.

Genre classification will be valuable when there are some interesting facts or problems, finding the most specific song which has been listened to many times.

Genre classification is important for many real-world applications. As the quantity of music being released daily continues to sky-rocket, especially on internet platforms such as Soundcloud6 and Spotify7. Spotify releases Forty thousand tracks per day, which is the equivalent of 280,000 songs a week, or around 1.2 million tracks per calendar month. In a year, this volume would add up to a whopping 14.6 million. Many companies8 nowadays use music classification, either to place recommendations to their customers (for example Spotify, Soundcloud) or as a product (for example Shazam9). Determining the music genre is the first step in that direction. To classify songs in any given playlist or library by genre is an important functionality for any music streaming/purchasing service.

CHAPTER 1

INTRODUCTION

DatasetOverview

The dataset we will use is named the GTZAN genre collection dataset which is a very popular audio collection dataset. It contains approximately 1000 audio files that belong to 10 different classes. Each audio file is in .wav format (extension). The classes to which audio files belong are Blues, Hip-hop, classical, pop, Disco, Country, Metal, Jazz, Reggae, and Rock. You can easily find the dataset over Kaggle and can download it from here.

Libraries Installation

Before we move to load dataset and model building it's important to install certain libraries. In the last blog, we have used librosa to extract features. Now we will use the python speech feature library to extract features and to have a different taste. as well as to load the dataset in the WAV format we will use the scipy library so we need to install these two libraries.

```
!pip install python_speech_features
!pip install scipy
```

Preparing the Dataset

- Choosing the Dataset

For this paper, we have used the GTZAN Dataset which consists of 10 genres, each consisting of 100 audio tracks, each track having a duration of 30 seconds. The dataset consists of pre-classified genres, namely blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae and rock. The benefit of having the genres mentioned respective to each class of audio files is it makes it easier to label the audio files with their genre name, which will help in training the Deep Learning model.

- Pre-Processing

Before using the dataset as it is, we need to make a few tweaks to it, in order to obtain better results. Since our dataset is not very big, we can virtually increase the number of samples, by splitting the audio files into smaller samples, thereby increasing the number of samples. Now that we have more samples, we have more data for the model to work on.
For the sake of explanation, if we split the audio file vectors into 3 sub-vectors, each will have a duration of 10 seconds, and the total audio sample count triples, allowing us to operate on a greater number of samples.

- Storing the data in appropriate format

Audio files can not be directly operated on, as important features such as power spectral density coefficients give better insights into frequency peaks, dips, and the gap between two peaks, harmonics and much more. Hence, we split the audio data, then using python's Librosa Library convert the time domain signal to frequency domain, and also use Librosa to obtain its Mel Frequency Cepstral Coefficients and begin training the Deep Learning model.

PROJECT DETAILS AND IMPLEMENTAION

Hands on project

Step-1) Import required libraries

Head over to Jupyter notebook or newly created Kaggle notebook and first thing is to make necessary imports to play with the data.

```
import numpy as np
import pandas as pd
import scipy.io.wavfile as wav
from python_speech_features import mfcc
from tempfile import TemporaryFile import
os
import math
import pickle
import random
import operator
```

Step-2) Define a function to calculate distance between feature vectors, and to find neighbours.

We know how KNN works is by calculating distance and finding the K number of neighbours. To achieve this only for each functionality we will implement different functions. First, we will implement a function that will accept training data, current instances, and the required number of neighbours. It will find the distance of each point with every other point in training data and then we find all the nearest K neighbours and return all neighbours. to calculate the

```
#define a function to get distance between feature vectors and find neighbors def
getNeighbors(trainingset, instance, k):
    distances = []
    for x in range(len(trainingset)):
        dist = distance(trainingset[x], instance, k) + distance(instance,trainingset[x],k)
        distances.append((trainingset[x][2], dist))
```

distance of two-point we will implement a function after explaining some steps to make a project workflow simple

and understandable. .

```
    distances.sort(key=operator.itemgetter(1))
    neighbors = []
    for x in range(k):
        neighbors.append(distances[x][0])
    return neighbors
```

Step-3) Identify the class of nearest neighbours

Now we are having a list of neighbours and we need to find out a class that has the maximum neighbours count. So we declare a dictionary that will store the class and its respective count of neighbours. After creating the frequency map we sort the map in descending order based on neighbours count and return the first class.

Step-4) Model Evaluation

```
#function to identify the nearest neighbors
def nearestclass(neighbors):
    classVote = {}
```

```
def getAccuracy(testSet, prediction):
    correct = 0
```

```
        response = neighbors[x]
        if response in classVote:
            classVote[response] += 1
        else:
            classVote[response] = 1
```

we also require a function that evaluates a model to check the accuracy and performance of the algorithm we build. so we will build a function that is a fairly simple accuracy calculator which says a total number of correct predictions divided by a total number of predictions.

```
    for x in range(len(testSet)):
        if testSet[x][-1] == prediction[x]:
            correct += 1
    return 1.0 * correct / len(testSet)
```

Step-5) Feature Extraction

you might be thinking like we have implemented the model and now we will extract features from data. So as we are using KNN Classifier and we have only implemented the algorithm from scratch to make you understand how the project runs and till now I hope you have a 70 percent of idea how the project will work. So now we will load the data from all the 10 folders of respective categories and extract features from each audio file and save the extracted features in binary form in DAT extension format.

Now we do not have to implement all these steps separately, MFCC brings all these for us which we have already imported from the python speech feature library. we will iterate through each category folder, read the audio file, extract the MFCC feature, and dump it in a binary file using the pickle module. I suggest always using try-catch while loading huge datasets to understand, and control if any exception occurs.

```
directory = '../input/gtzan-dataset-music-genre-classification/Data/genres_original' f =
open("mydataset.dat", "wb")
i = 0
for folder in os.listdir(directory):
    #print(folder)
    i += 1
    if i == 11:
        break
    for file in os.listdir(directory+"/"+folder):
        #print(file)
        try:
            (rate, sig) = wav.read(directory+"/"+folder+"/"+file)
            mfcc_feat = mfcc(sig, rate, winlen = 0.020, appendEnergy=False)
            covariance = np.cov(np.matrix.transpose(mfcc_feat)) mean_matrix =
            mfcc_feat.mean(0)
```

```
            feature = (mean_matrix, covariance, i)
            pickle.dump(feature, f)
        except Exception as e:
            print("Got an exception: ", e, 'in folder: ', folder, ' filename: ', file)
f.close()
```

Step-6) Train-test split the dataset

Now we have extracted features from the audio file which is dumped in binary format as a filename of my dataset. Now we will implement a function that accepts a filename and copies all the data in form of a dataframe. After that based on a certain threshold, we will randomly split the data into train and test sets because we want a mix of different genres in both sets. There are different approaches to do train test split. here I am using a random module and running a loop till the length of a dataset and generate a random fractional number between 0-1 and if it is less than 66 then a particular row is appended in the train test else in the test set.

```
dataset = []


def loadDataset(filename, split, trset, teset):
    with open('my.dat','rb') as f:
        while True:
            try:
                dataset.append(pickle.load(f))
            except EOFError:
                f.close()
                break
    for x in range(len(dataset)):
        if random.random() < split:
            trset.append(dataset[x])
        else:
            teset.append(dataset[x])
```

```
testSet = []
loadDataset('my.dat', 0.68, trainingSet, testSet)
```

Step-7) Calculate the distance between two instance

This function we have to implement at the top to calculate the distance between two points but to explain to you the complete workflow of the project, and algorithm I am explaining the supporting function after the main steps. But you need to add the function on top. So, the function accepts two data points(X, and y coordinates) to calculate the actual distance between them. we use the numpy linear algebra package which provides a low-level implementation of standard linear algebra. So we first find the dot product between the X-X and Y-Y coordinate of both points to know the actual distance after that we extract the determinant of the resultant array of both points and get the distance.

```
def distance(instance1, instance2, k):
    distance = 0
    mm1 = instance1[0]
    cm1 = instance1[1]
    mm2 = instance2[0]
    cm2 = instance2[1]
    distance = np.trace(np.dot(np.linalg.inv(cm2), cm1))
    distance += (np.dot(np.dot((mm2-mm1).transpose(), np.linalg.inv(cm2)), mm2-mm1))
    distance += np.log(np.linalg.det(cm2)) - np.log(np.linalg.det(cm1))
    distance -= k
    return distance
```

Step-8) Training the Model and making predictions

The step has come you all were waiting to feed the data to KNN algorithms and make all predictions and receive accuracy on the test dataset. This step code seems to be large but it is very small because we are following a functional programming approach in a stepwise manner so we only need to call the functions. The first is to get neighbors, extract class and check the accuracy of the model.

```
# Make the prediction using KNN(K nearest Neighbors)
length = len(testSet)
```

```
predictions = []
for x in range(length): predictions.append(nearestclass(getNeighbors(trainingSet,
    testSet[x], 5)))

accuracy1 = getAccuracy(testSet, predictions)
print(accuracy1)
```

Step-9) Test the Classifier with the new Audio File

Now we have implemented and trained the model and it's time to check for the new data to check how much our model is compliant in predicting the new audio file. So we have all the labels (classes) in numeric form, and we need to check the class name so first, we will implement a dictionary where the key is numeric label and value is the category name.

```
from collections import defaultdict
results = defaultdict(int)

directory = "../input/gtzan-dataset-music-genre-classification/Data/genres_original"

i = 1
for folder in os.listdir(directory):
    results[i] = folder
```

Now we can predict a new audio file and receive a label for it and print the name of the category using the result dictionary.

Future Scope

As our communication becomes increasingly influenced and dependent on technology, and releasing music becomes easier, many people will use music as a means not to create a career – but as a social endeavor, or form of communication. Furthermore, music as a social endeavor will increase as the means of making music becomes easier.

An automatic genre classification algorithm could greatly increase efficiency for music databases such as AllMusic. It could also help music recommender systems and playlist generators that companies like Spotify and Pandora use. It's also a really fun problem to play around with if you love music and data!

Accuracy

On simple analysis we realize that using MFCC will be better as the classifier input. So now w create the model according to the approach we desire. For this paper, where MFCC values along with their pre classified genres as the labels are the inputs, we will train five types of models which

are best suited for the case:

Convolutional Neural Networks, Recurrent Neural Network, K Nearest Neighbors, Naïve Bayes Classifier and Support Vector Machine models and their performance will be noted.

Accuracy calculation for performance is as follows:
Accuracy % = 100 * Correct / Total Guesses

```
# Make the prediction using KNN(K nearest Neighbors)
length = len(testSet)
predictions = []
for x in range(length):
    predictions.append(nearestclass(getNeighbors(trainingSet, testSet[x], 5)))

accuracy1 = getAccuracy(testSet, predictions)
print(accuracy1)
```

```
0.7066666666666667
```

Result

As we can see most of the techniques fare comparatively well, some even go above 70%. The main thing to focus on is that the GTZAN Dataset does not contain enough number of audio samples to classify well enough, so having more data is necessary.

The tabulation of performance shows that relatively the Support Vector Machine and the Convolution Neural Network Models performed much better than the others. However, we must consider that our dataset was relatively much smaller than the actual number of songs that are present, hence increasing the dataset size, number of genres, usage of different features of the data may account to different Deep Learning Models outperforming these.

We also learn that by taking fundamental features from audio wav files, it is possible to learn about the features of music, and even possible to classify its genre and possible predict the future music just be having sufficient of its past values.

```
pred = nearestclass(getNeighbors(dataset, feature, 5))
print(results[pred])
```

```
pop
```

References

- https://www.a_nalyticssteps.com/blogs/music-genre-classification-using-machine-learning
- https://www.a_nalyticsvidhya.com/blog/2022/03/music-genre-classification-project-using-machine-learning-techniques/
- https://towardsdatascience.com/musical-genre- classification-with-convolutional-neural-networks- ff04f9601a74