VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

WINTER SEMESTER 2020-2021

CSE 3013

ARTIFICIAL INTELLIGENCE

J-COMPONENT

PROJECT REPORT

FACULTY: RAJAKUMAR K

MEMBERS:

SHUBH ALMAL (19BCE2130)
APOORVA REDDY (19BCE2196)
RIYA GUPTA (19BCE2072)

TOPIC: MUSIC GENRE CLASSIFICATION

ABSTRACT

A music genre is a conventional category that identifies some pieces of music as belonging to a shared tradition or set of conventions. It is to be distinguished from musical form and musical style.

Categorizing music files according to their genre is a challenging task in the area of music information retrieval (MIR). Automatic music genre classification is important to obtain music from a large collection. It finds applications in the real world in various fields like automatic tagging of unknown piece of music (useful for apps like Saavn, Wynk etc.).

Musical genres are categorical labels created by humans to characterize pieces of music. A musical genre is characterized by the common characteristics shared by its members. These characteristics typically are related to the instrumentation, rhythmic structure, and harmonic content of the music. Genre hierarchies are commonly used to structure the large collections of music available on the Web.

Machine Learning techniques have proved to be quite successful in extracting trends and patterns from the large pool of data. Since manually classifying each track of a large music database according to their genre is a tedious task, Machine Learning Techniques to perform Automatic Music Genre Classification are used. It can assist or replace the human user in this process and would be a valuable addition to music information retrieval systems.

There are various methods to perform classification on this dataset. Some of these approaches are:

Multiclass support vector machines
K-means clustering
K-nearest neighbors
Convolutional neural networks

A lot of projects have worked only on K-nearest neighbors but we will work on CNN model too.

INTRODUCTION

The classification of genres is a critical undertaking with several real-world applications. The amount of music released on a daily basis continues to rise, particularly on internet platforms like Saavn, Soundcloud, Spotify and Wynk. As the requirement for precise meta-data for database management and search/storage grows, so does the requirement for accurate meta-data. The ability to rapidly identify songs in any given playlist or collection by genre is a critical feature for any music streaming/purchasing business, and the statistical analysis potential provided by accurate and full labelling of music and audio is virtually endless.

Convolutional neural networks have been demonstrated to be extremely accurate music genre classifiers, with outstanding results demonstrating both the complexity afforded by several layers and the capacity of convolutional layers to accurately identify patterns within images. The human capability for genre classification has been greatly exceeded by these results. Many of the studies that used CNNs compared their models to other machine learning approaches including k-NN, mixtures of Gaussians, and SVMs, and CNNs outperformed them all.

LITERATURE SURVEYS

Paper 1: Music Genre Classification using Machine Learning Techniques by Hareesh Bahuleyan

The authors compare the performance of two types of models in this research. The first is a deep learning strategy in which a CNN model is trained end-to-end to predict an audio signal's genre label simply based on its spectrogram. The second method makes use of hand-crafted characteristics in both the temporal and frequency domains. It use these features to train four typical machine learning classifiers and compare their performance, presenting an AUC value of 0.894 for an ensemble classifier that integrates the two proposed techniques in the experiments on the Audio data set. CNN-based deep learning models outperformed feature-engineered models, according to the findings. It also showed that combining the CNN and XGBoost models yielded positive results.

Paper 2: Automatic Music Genre Classification Using Bass Lines by Umut Simsekli

On a small size standard MIDI database, “Melodic Interval Histograms” are employed as features, and k-nearest neighbour classifiers are used and compared with SVMs. An unique distance metric, perceptually weighted Euclidean distance (PWED) was suggested, in addition to existing distance metrics for k-nearest neighbours (Euclidean, symmetric Kullback-Leibler, earth mover's, normalised compression distances). To investigate the overall performance of all k-NN configurations, the root accuracies were first calculated using all configurations, then the leaf accuracies were calculated as if the root labels had been accurately classified. With k- nearest neighbour classifiers, the highest classification accuracy (84%) is achieved, and the increased utility of the novel measure is demonstrated in the trials.

Paper 3: An Evaluation of Convolutional Neural Networks for Music Classification Using Spectrograms by Yandre M. G. Costa, Luiz S. Oliveira and Carlos N. Silla Jr

This paper used three music databases with different properties to test the Convolutional Neural Network for music content characterization. The experiments have demonstrated that the CNN outperforms other classifiers. On the African music database, the CNN outperformed all published results, while in the LMD database, the combination of a CNN and an SVM trained using RLBP achieved a recognition rate of 92 percent, which, to their knowledge, the best result (using the artist filter) on their dataset. The CNN did not improve the state-of-the-art in the ISMIR 2004 database, but when compared to all individual classifiers, it outperformed them all. All of these findings support their hypothesis that deep learning-based representation learning is a feasible alternative to hand-crafted feature design in automatic feature engineering for music content characterisation, which has dominated in recent years.

Paper 4: Music Genre Classification Using Convolutional Neural Network by Qiuqiang Kong, Xiaohui Feng and Yanxiong Li

This paper presents an algorithm based on spectrograms and convolutional neural networks (CNN). The spectrogram contains additional details of music components such as pitch, flux, and so on, as compared to MFCC. They convolve a spectrogram with a feature detector as a filter to produce four

feature maps that can catch spectrogram trends on both a temporal and frequency scale. Then, to reduce dimension and improve resistance to pitch and tempo translation, a sub-sample layer is added. Finally, a multi-layer perceptron (MLP) classifier is used to connect the extracted high-level characteristics. The proposed features achieve a classification accuracy of 72.4 percent on the Tzanetakis dataset, which outperforms MFCC.

Paper 5: Music Genre Classification Using MIDI and Audio Features by Zehra Cataltepe, Yusuf Yaslan, and Abdullah Sonmez

In this paper, a normalised compression distance (NCD) and a 10-nearest neighbour (10 NN) classifier to classify genres in MIDI files were used. Features were used at varying sample rates and sizes, as well as LDC and KNN classifiers, to classify genres from MIDI files. They used alternative systems of majority voting to integrate the 12 different classifiers. The authors discovered that using a majority vote enhanced categorization accuracy. The categorization accuracy for MIDI or audio alone was significantly lower than the expected results. The study concluded that the use of multiple classifiers enhanced genre classification.

Paper 6: Improved Music Genre Classification with Convolutional Neural Networks by Weibin Zhang, Wenkang Lei, Xiangmin Xu and Xiaofeng Xing

In this paper, the authors propose two methods for improving music genre classification with convolutional neural networks: using shortcut connections to skip one or more layers, a method inspired by residual learning; and combining max- and average-pooling to provide more statistical information to higher level neural networks. The CNN's input is merely the audio signal's short-time Fourier transformations. The CNN's output is passed into another deep neural network that does classification. The preliminary experimental results on the GTZAN data set demonstrate that the aforementioned two strategies, particularly the first, can effectively increase classification accuracy by comparing two different network topologies.

Paper 7: Music Feature Maps with Convolutional Neural Networks for Music Genre Classification by Christine Senac, Thomas Pellegrini, Florian Mouret and Julien Pinquier

As inputs to a CNN, the authors decided to employ a map of eight musical features. These characteristics were chosen from a broader set evaluated in an earlier study along the dimensions of dynamics, timbre, and tone. They used CNNs that had been trained in such a way that the filter dimensions (adjusted for their needs) could be interpreted in time and frequency. The results demonstrate the use of their eight music features: global accuracy of 89.6% versus 87.8% for 513 frequency bins in a spectrogram. On the GTZAN database, late score fusion between systems based on both feature types achieves 91 percent accuracy. This success can be attributed to two factors: first, it avoids the more or less difficult extraction of carefully engineered audio features; second, the deep learning hierarchical topology is beneficial for musical analysis because music is hierarchical in frequency and time, and relationships between musical events in the time domain.

Paper 8: Audio Music Genre Classification Using Different Classifiers and Feature Selection Methods by Yusuf Yaslan and Zehra Cataltepe

To classify the genre of a given piece of music, the authord of this paper analyzed ten different classifiers on different audio feature sets, including Fisher, LDC, QDC, UDC, NBC, PDC, KNN, KNN1, KNN3, and KNN5. They also analysed the performance of feature sets created by dimensionality reduction strategies for each classifier and then tried mixing multiple classifiers to improve classification accuracy. They initially acquired a test genre classification accuracy of roughly 79.64.2 percent using a series of various classifiers on a 10 genre set of 1000 music samples. This score outperforms the previous best of 71.17.3 percent, which was published on this data set. The authors can also achieve a classification accuracy of 80 percent by reducing dimensionality or mixing several classifiers. On the results of their study, the authors theorize that the best feature set appears to be dependent on the classifier utilised.

Paper 9: Automatic Genre Classification Using Large High-Level Musical Feature Sets by Cory McKay and Ichiro Fujinaga

This study describes a system for extracting 109 musical elements from symbolic recordings (MIDI) and categorising them by genre. Instrumentation, texture, rhythm, dynamics, pitch statistics, melody, and chords are among the features used here. Different sets of features are used at different levels of the hierarchy to perform the categorization. Genetic algorithms are employed to select which features are used at each stage and their relative weightings.

A unique ensemble of feedforward neural networks and k-nearest neighbour classifiers is used for classification. The effect of increasing the number of candidate features on classification performance is investigated in order to empirically highlight the relevance of using a vast diversity of musically significant information. To test the system's performance under different settings, two different sized hierarchies are adopted. For a hierarchical taxonomy consisting of 9 leaf genres, classification success rates of 98 percent for root genres and 90 percent for leaf genres were found.

Paper 10: Music Genre Classification by Derek A. Huang, Arianna A. Serafini and Eli J. Pugh

A RBF kernel support vector machine, k-nearest neighbours, a simple feed-forward network, and eventually an advanced convolutional neural network were used in the experiments. The authors experimented with both raw amplitude data and processed mel-spectrograms of that raw amplitude data as input to their algorithms. They then generated a predicted genre from a list of ten popular music genres. All of their models performed better once they converted the raw audio into mel-spectrograms, with convolutional neural network outperforming human accuracy. Although CNN took the longest to train, the increased accuracy outweighs the additional computing cost.

PROPOSED WORK

Our project makes a basic attack on the music genre classification problem. We used a single feature (MFCCs) throughout this project. This gives a fair comparison of learning algorithms, exploring the effectiveness of different features would help to determine which machine learning stack does best in music classification.

Since genre classification between fairly different genres is quite successful, it makes sense to attempt finer classifications. The exact same techniques used in this project could be easily extended to classify music based on any other labelling, such as artist. In addition, including additional metadata text features such as album, song title, or lyrics could allow us to extend this to music mood classification as well.

IMPLEMENTATION

We have developed the music genre classification using Convolutional Neural Network (CNN), using 3 convolution layers, each with its own max pool and Batch Normalization , feeding into 3 fully connected layers with ReLU activation, softmax output, and sparse_categorical_crossentropy loss. Dropout layer has been implemented in the 3rd convolutional layer after there was an overfitting in the training dataset. Dropout works by randomly setting the outgoing edges of hidden units (neurons that make up hidden layers) to 0 at each update of the training phase. We used sparse categorical cross entropy because our genre classes are mutually exclusive as we trained the model with 400 epochs.

Before we use the dataset we have extract the features from it, so we will extractMel-frequency cepstral coefficients also known as MFCC are a feature widely used in automatic speech and speaker recognition.

MUSIC GENRE CLASSIFIER

We have developed music genre classifier using Convolutional Neutral Network(CNN), It will take the input .wav file and classify it into its respective genre

There are 10 genre as follows :

Blue

Country

Disco

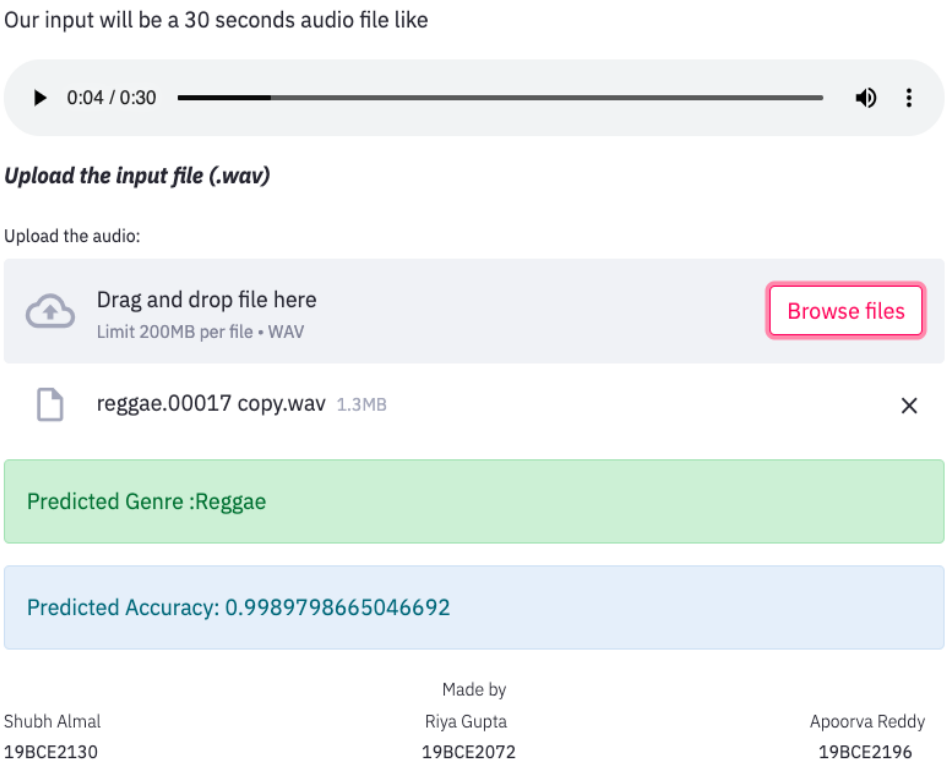
Hiphop

Jazz

Metal

Pop

Reggae




```
# Import Libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import os, json, math, librosa
import IPython.display as ipd
import librosa.display
import tensorflow as tf
import tensorflow.keras as keras
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Conv2D
import sklearn.model_selection as sk
from sklearn.model_selection import train_test_split
```

```
# Collect the dataset from the path directory
MUSIC = '/Users/apoorva/Desktop/CODES/Data/genres_original/'
music_dataset = []
genre_target = []
for root, dirs, files in os.walk(MUSIC):
    for name in files:
        filename = os.path.join(root, name)
        if filename != '/data/genres_original/jazz/jazz.00054.wav':
            music_dataset.append(filename)
```

```
            genre_target.append(filename.split("/")[5])

# sample audio file
audio_path = music_dataset[500]
ipd.Audio(audio_path)


DATASET_PATH = '/Users/apoorva/Desktop/CODES/Data/genres_original/'
JSON_PATH = "music_genre.json"
SAMPLE_RATE = 22050
TRACK_DURATION = 30
SAMPLES_PER_TRACK = SAMPLE_RATE * TRACK_DURATION

# Convert the audio file into 3D Array of features
def save_mfcc(dataset_path, json_path, num_mfcc=13, n_fft=2048, hop_length=512,
num_segments=5):

    data = {
        "mapping": [],
        "labels": [],
        "mfcc": []
    }

    samples_per_segment = int(SAMPLES_PER_TRACK / num_segments)
    num_mfcc_vectors_per_segment = math.ceil(samples_per_segment / hop_length)


    for i, (dirpath, dirnames, filenames) in enumerate(os.walk(dataset_path)):

        if dirpath is not dataset_path:

            semantic_label = dirpath.split("/")[-1]
            data["mapping"].append(semantic_label)
            print("\nProcessing: {}".format(semantic_label))

            for f in filenames:

                file_path = os.path.join(dirpath, f)

                if file_path !=
'/Users/apoorva/Desktop/CODES/Data/genres_original/jazz/jazz.00054.wav':

                    signal, sample_rate = librosa.load(file_path, sr=SAMPLE_RATE)


                    for d in range(num_segments):

                        start = samples_per_segment * d
                        finish = start + samples_per_segment
```

```
                    mfcc = librosa.feature.mfcc(signal[start:finish], sample_rate,
n_mfcc=num_mfcc, n_fft=n_fft, hop_length=hop_length)
                    mfcc = mfcc.T

                    if len(mfcc) == num_mfcc_vectors_per_segment:
                        data["mfcc"].append(mfcc.tolist())
                        data["labels"].append(i-1)
                        print("{}, segment:{}".format(file_path, d+1))

    with open(json_path, "w") as fp:
        json.dump(data, fp, indent=4)

save_mfcc(DATASET_PATH, JSON_PATH, num_segments=6)


DATA_PATH = "./music_genre.json"


def load_data(data_path):

    with open(data_path, "r") as fp:
        data = json.load(fp)

    X = np.array(data["mfcc"])
    y = np.array(data["labels"])
    z = np.array(data['mapping'])
    return X, y,z

# Split the dataset into training,validating,testing
def prepare_datasets(test_size, validation_size):

    # load data
    X, y,z = load_data(DATA_PATH)

    # create train, validation and test split
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size)
    X_train, X_validation, y_train, y_validation = train_test_split(X_train, y_train,
test_size=validation_size)

    # add an axis to input sets
    X_train = X_train[..., np.newaxis]
    X_validation = X_validation[..., np.newaxis]
    X_test = X_test[..., np.newaxis]

    return X_train, X_validation, X_test, y_train, y_validation, y_test,z


# Build CNN Model
def build_model(input_shape):
```

```
    model = keras.Sequential()

    # 1st conv layer
    model.add(keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=input_shape))
    model.add(keras.layers.MaxPooling2D((3, 3), strides=(2, 2), padding='same'))
    model.add(keras.layers.BatchNormalization())

    # 2nd conv layer
    model.add(keras.layers.Conv2D(32, (3, 3), activation='relu'))
    model.add(keras.layers.MaxPooling2D((3, 3), strides=(2, 2), padding='same'))
    model.add(keras.layers.BatchNormalization())

    # 3rd conv layer
    model.add(keras.layers.Conv2D(32, (2, 2), activation='relu'))
    model.add(keras.layers.MaxPooling2D((2, 2), strides=(2, 2), padding='same'))
    model.add(keras.layers.BatchNormalization())

    # flatten output and feed it into dense layer
    model.add(keras.layers.Flatten())
    model.add(keras.layers.Dense(64, activation='relu'))
    model.add(keras.layers.Dropout(0.8))

    # output layer
    model.add(keras.layers.Dense(10, activation='softmax'))

    return model

# Prediction
def predict(model, X, y):

    X = X[np.newaxis, ...]

    prediction = model.predict(X)

    predicted_index = np.argmax(prediction, axis=1)

    target = z[y]
    predicted = z[predicted_index]

    print("Target: {}, Predicted label: {}".format(target, predicted))


X_train, X_validation, X_test, y_train, y_validation, y_test,z = prepare_datasets(0.25, 0.2)

input_shape = (X_train.shape[1], X_train.shape[2], 1)
model = build_model(input_shape)

optimiser = keras.optimizers.Adam(learning_rate=0.0001)
```

```
model.compile(optimizer=optimiser,
            loss='sparse_categorical_crossentropy',
            metrics=['accuracy'])

model.summary()


history = model.fit(X_train, y_train, validation_data=(X_validation, y_validation),
batch_size=32, epochs=30)

model.fit(X_train, y_train, validation_data=(X_validation, y_validation), batch_size=32,
epochs=30)

model.fit(X_train, y_train, validation_data=(X_validation, y_validation), batch_size=32,
epochs=30)

model.fit(X_train, y_train, validation_data=(X_validation, y_validation), batch_size=32,
epochs=30)

model.fit(X_train, y_train, validation_data=(X_validation, y_validation), batch_size=32,
epochs=30)

model.fit(X_train, y_train, validation_data=(X_validation, y_validation), batch_size=32,
epochs=30)

model.fit(X_train, y_train, validation_data=(X_validation, y_validation), batch_size=32,
epochs=30)

model.fit(X_train, y_train, validation_data=(X_validation, y_validation), batch_size=32,
epochs=10)

model.fit(X_train, y_train, validation_data=(X_validation, y_validation), batch_size=32,
epochs=10)

model.fit(X_train, y_train, validation_data=(X_validation, y_validation), batch_size=32,
epochs=10)

model.fit(X_train, y_train, validation_data=(X_validation, y_validation), batch_size=32,
epochs=10)

model.fit(X_train, y_train, validation_data=(X_validation, y_validation), batch_size=32,
epochs=100)


model_json = model.to_json()
with open("music_genre(80).json", "w") as json_file:
    json_file.write(model_json)

model.save_weights("music_genre(80).h5")
print("Saved model to disk")
```

```
X_to_predict = X_test[500]
y_to_predict = y_test[500]

predict(model, X_to_predict, y_to_predict)


from keras.models import model_from_json

json_file = open('music_genre(80).json', 'r')
model = json_file.read()
json_file.close()
loaded_model = model_from_json(model)

loaded_model.load_weights("music_genre(80).h5")
print("Loaded model from disk")

optimiser = keras.optimizers.Adam(learning_rate=0.0001)
loaded_model.compile(optimizer=optimiser,
            loss='sparse_categorical_crossentropy',
            metrics=['accuracy'])

loaded_model

tf.keras.models.save_model(loaded_model,'music_model.hdf5')

WEB APP CREATED USING STREAMLIT

import streamlit as st
from PIL import Image
import tensorflow as tf
import os, json, math, librosa
import numpy as np

img = Image.open("/Users/apoorva/Desktop/images.jpeg")
st.image(img,width=700)
st.title("MUSIC GENRE CLASSIFIER")

st.markdown("We have developed music genre classifier using Convolutional Neutral
Network(CNN), It will take the input .wav file and classify it into its respective genre")

st.markdown("There are 10 genre as follows :")
genre_list = ['Blue','Country','Disco','Hiphop','Jazz','Metal','Pop','Reggae','Rock','Classical']

for i in genre_list:
        st.markdown('**'+i+'**')

st.write("Our input will be a 30 seconds audio file like")
audio_file =
open('/Users/apoorva/Desktop/CODES/Data/genres_original/disco/disco.00002.wav','rb')
```

```
audio_bytes = audio_file.read()
st.audio(audio_bytes,format='audio/wav')

st.markdown("**_Upload the input file (.wav)_**")

audio_file = st.file_uploader("Upload the audio:",type=["wav"])
if audio_file is None:
        st.error("Upload a file!!")

@st.cache(allow_output_mutation=True)
def load_model():

model=tf.keras.models.load_model('/Users/apoorva/Desktop/CODES/Notebook/music_mo
del.hdf5')
  return model

with st.spinner('Model is being loaded..'):
  model=load_model()

#DATASET_PATH = audio_file
JSON_PATH = "music.json"
SAMPLE_RATE = 22050
TRACK_DURATION = 30
SAMPLES_PER_TRACK = SAMPLE_RATE * TRACK_DURATION

def save_mfcc(file_path, json_path, num_mfcc=13, n_fft=2048, hop_length=512,
num_segments=5):


    data = {
        "mapping": [],
        "labels": [],
        "mfcc": []
    }

    samples_per_segment = int(SAMPLES_PER_TRACK / num_segments)
    num_mfcc_vectors_per_segment = math.ceil(samples_per_segment / hop_length)


    signal, sample_rate = librosa.load(file_path, sr=SAMPLE_RATE)



    for d in range(num_segments):
        start = samples_per_segment * d
        finish = start + samples_per_segment


        mfcc = librosa.feature.mfcc(signal[start:finish], sample_rate, n_mfcc=num_mfcc,
n_fft=n_fft, hop_length=hop_length)
```

```
            mfcc = mfcc.T


            if len(mfcc) == num_mfcc_vectors_per_segment:
                data["mfcc"].append(mfcc.tolist())
                #data["labels"].append(i-1)



        with open(json_path, "w") as fp:
            json.dump(data, fp, indent=4)

if audio_file is not None:
    save_mfcc(audio_file, JSON_PATH, num_segments=6)
else:
            st.warning("Upload a file!!")


DATA_PATH = "./music.json"


def load_data(data_path):

    with open(data_path, "r") as fp:
        data = json.load(fp)

    X = np.array(data["mfcc"])
    #y = np.array(data["labels"])
    z = np.array(data['mapping'])
    return X,z

X,z = load_data(DATA_PATH)

X = X[..., np.newaxis]

def predict(model, X):



    prediction = model.predict(X)

    predicted_index = np.argmax(prediction, axis=1)


    #predicted = z[predicted_index]

    #print("Predicted label: {}".format(predicted))

    return prediction
```

```
p = predict(model,X)

music_dict =
{0:'Pop',1:'Metal',2:'Disco',3:'Blues',4:'Reggae',5:'Classical',6:'Rock',7:'HipHop',8:'Country',
9:'Jazz'}

for key,value in music_dict.items():

    if p[0][key]>0.7:
        st.success("Predicted Genre :" + value)
        st.info("Predicted Accuracy: {}".format(p[0][key]))
```

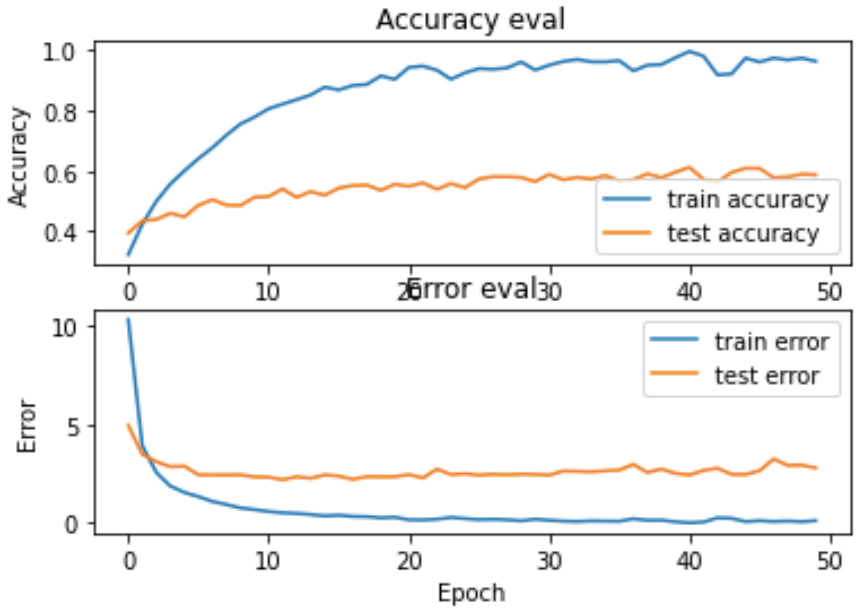
RESULT

The CNN had a success rate of 90% when identifying some genre during the testing time. The main quantitative metric which we used to judge our model is accuracy (that is, percentage of predicted labels which matched their true labels). We selected our hyperparameters based of empirical results and industry standards. For instance, choosing 3 by 3 window for our first convolution window was a result of seeing a similar window size work in other academic results, and then fine tuning to meet our data-specific needs. We chose to use Adam optimization for a few reasons. Working with audio data over 2 dimensions cause sparse gradient problems. Adam mitigates the sparse gradient problem by maintaining a per-parameter learning rate, and mitigates the noise problem by basing updates on a weighted average of recent updates (momentum). With Adam our models trained more quickly and didn't plateau as early.

```
47/47 - 1s - loss: 0.3572 - accuracy: 0.9099
113/113 - 3s - loss: 0.3910 - accuracy: 0.9118

Test accuracy: 0.9098798632621765

Train accuracy: 0.9118219614028931
```



REFERENCES

[1] Bahuleyan, Hareesh. "Music genre classification using machine learning techniques." *arXiv preprint arXiv:1804.01149* (2018).

[2] Şimşekli, Umut. "Automatic music genre classification using bass lines." *2010 20th International Conference on Pattern Recognition*. IEEE, 2010.

[3] Costa, Yandre MG, Luiz S. Oliveira, and Carlos N. Silla Jr. "An evaluation of convolutional neural networks for music classification using spectrograms." *Applied soft computing* 52 (2017): 28-38.

[4] Kong, Qiuqiang, Xiaohui Feng, and Yanxiong Li. "Music genre classification using convolutional neural network." *Proc. of Int. Society for Music Information Retrieval Conference (ISMIR)*. 2014.

[5] Cataltepe, Zehra, Yusuf Yaslan, and Abdullah Sonmez. "Music genre classification using MIDI and audio features." *EURASIP Journal on Advances in Signal Processing* 2007 (2007): 1-8.

[6] Zhang, Weibin, et al. "Improved Music Genre Classification with Convolutional Neural Networks." *Interspeech*. 2016.

[7] Senac, Christine, et al. "Music feature maps with convolutional neural networks for music genre classification." *Proceedings of the 15th International Workshop on Content-Based Multimedia Indexing*. 2017.

[8] Yaslan, Yusuf, and Zehra Cataltepe. "Audio music genre classification using different classifiers and feature selection methods." *18th International Conference on Pattern Recognition (ICPR'06)*. Vol. 2. IEEE, 2006.

[9] McKay, Cory, and Ichiro Fujinaga. "Automatic Genre Classification Using Large High-Level Musical Feature Sets." *ISMIR*. Vol. 2004. No. 2004. 2004.

[10] Huang, Derek A., Arianna A. Serafini, and Eli J. Pugh. "Music Genre Classification."