Project Report on

'MUSIC GENRE CLASSIFICATION'

Submission to **THE ROBOTICS CLUB** as a part of

DEEP LEARNING

TECHNICAL ADVISORY BOARD'24

THE ROBOTICS CLUB

Integrating Knowledge....

THE ROBOTICS CLUB–SNIST

SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY(AUTONOMOUS)

(Affiliated to JNTU University, Hyderabad)Yamnapet,

Ghatkesar, Hyderabad-501301.

2023

CERTIFICATE

This is the project work titled **MUSIC GENRE CLASSIFICATION** by **Manoj Reddy, Surya Teja, Jerry K Paul, Dheeraj, Manvitha, Tejaswi, Srinath, Harini** under the guidance of **SATHWIKA AMARANAYANI** for the recruitment into the **TECHNICAL ADVISORY BOARD** and is a record of the project work carried out by them during the year 2023-2024 as part of **TAB** under the supervision of

BANGARU RAKESH

TAB Chairman

SAATHVIKA REVALLA

TAB Vice-Chairman

Mr. N.V.V.S NARAYANA

The President of THE ROBOTICS CLUB

Dr. A. Purushotham

Technical Advisor

Mechanical Department

DECLARATION

The project work reported in the present thesis titled "**MUSIC GENRE CLASSIFICATION**" is a record work done by Deep Learning in **THE ROBOTICS CLUB** as a part of **TECHNICAL ADVISORY BOARD.**

No part of the thesis is copied from books/ journals/ Internet and wherever the portion is taken, the same has been duly referred in the text. The report is based on the project work done entirely by Deep Learning team and not copied from any other source.

ACKNOWLEDGMENT

This project report is the outcome of the efforts of many people who have driven our passion to explore into implementation of ‘**MUSIC GENRE CLASSIFICATION’**. We have received great guidance, encouragement, and support from them and have learned a lot because oftheir willingness to share their knowledge and experience. Primarily, we would like to express our gratitude to our mentor ‘**SATHWIKA AMARANAYANI’** Her guidance has been of immense help in surmounting various hurdles along the path of our goal.

We thank our TAB heads **RAKESH BANGARU**, **SAATHVIKA REVALLA, DEEKSHITH DOGIPARTHI** and **BHARGAV GUMMADELLY** for being with us till the end of the project completion.

We also thank our faculty advisor **Dr. A. PURUSHOTTAM**, Professor, Mechanical Department, who encouraged us during this project by rendering his help when need.

Contents

ABSTRACT
THE ROBOTICS CLUB
DEEP LEARNING

(Music Genre Classification Using Deep Learning)

The Problem Statement: The task of music genre classification involves automatically categorizing music tracks into predefined genres based on their audio features. This problem is significant for various applications in the music industry, such as recommendation systems, music cataloguing, and automated playlist generation. Traditional methods for genre classification often rely on manual feature extraction and shallow machine learning models, which may not effectively capture the complex patterns in audio data. The objective of this project is to develop a robust and accurate deep learning model capable of classifying music tracks into genres using raw audio data.

The Approach: To develop a robust and accurate deep learning model for music genre classification, we will follow a structured approach consisting of data preparation, model design, training, evaluation, and fine-tuning. Use the GTZAN dataset, which includes 1000 audio tracks evenly distributed across 10 genres (blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, rock). Load audio tracks using a library such as Librosa. Convert raw audio into Mel-spectrograms to capture time-frequency representations. Accept Mel-spectrogram or MFCC images. Stack multiple convolutional layers with ReLU activation functions. Use Max Pooling layers to reduce dimensionality while retaining essential features. Apply after each convolutional layer to normalize outputs and accelerate convergence. Add LSTM layers to capture sequential dependencies and temporal patterns in the audio data. Optionally use bidirectional LSTMs to capture dependencies in both forward and backward directions. Use fully connected layers to integrate features from CNN and LSTM layers. Use a softmax activation function to output probability distributions over the 10 genres. Analyze the confusion matrix to understand misclassifications and identify areas for improvement. Use TensorFlow or PyTorch for building and training the neural network models. Use Librosa for audio processing and feature extraction. Use Scikit-learn for evaluation metrics and cross-validation. Deploy the trained model as a web service or integrate it into a music application for real-time genre classification.

TITLE OF THE PROJECT: MUSIC GENRE CLASSIFICATION

MUSIC GENRE CLASSIFICATION

MEMBERS OF THE ROBOTICS CLUB SREENIDHI INSTITUTE

OF SCIENCE AND TECHNOLOGY, (SECOND YEAR)

GHATKESAR, HYDERABAD (SNIST)

Abstract

Music genre classification is a crucial task in the field of music information retrieval, enabling various applications such as music recommendation, playlist generation, and automated tagging. Traditional methods rely on handcrafted features and shallow models, which often fall short in capturing the intricate patterns present in audio signals. This project explores the use of deep learning techniques to improve the accuracy and robustness of music genre classification. By leveraging Convolutional Neural Networks (CNNs) and we aim to build a model that can automatically learn features from raw audio data and classify music into predefined genres.

Keywords: Convolutional Neural Network, Deep Learning

1. INTRODUCTION

Music genre classification is a fascinating and challenging task in the field of music information retrieval (MIR). It involves automatically categorizing music tracks into predefined genres based on their audio features. Deep learning, a subset of machine learning, has shown significant promise in this domain due to its ability to learn complex patterns and representations from data. Music genre classification aims to automate the identification of the genre of a music track. Traditional approaches relied on hand-crafted features and signal processing techniques.

In the context of music genre classification, deep learning models can automatically extract relevant features from audio signals without the need for manual feature engineering. This leads to improved accuracy and generalization across diverse music genres.

2. PROBLEM STATEMENT

Create a DL model to classify music tracks into different genres based on the input audio features. Model Architecture: Use a 1D CNN or a combination of CNN and LSTM to process the audio features.

3.LITERATURE SURVEY

Deep learning (DL) is a powerful machine learning field that has achieved considerable success in many research areas. Especially in the last decade, the-state-of-the-art studies on many research areas such as computer vision, object recognition, speech recognition, natural language processing were led to the awakening of the artificial intelligence from deep sleep. Nowadays, many researchers are trying to find solutions to many problems in various fields under the light of DL methods. In this study, we present important knowledge to guide about DL models and challenging topics which can be used in DL for researchers. We investigated DL studies which are made in the most popular and challenging fields such as Autonomous Vehicles, Natural Language Processing, Handwritten Character Recognition, Signature Verification, Voice and Video.

II.ARCHITECTURE

COMPONENTS REQUIRED

A. SOFTWARE COMPONENTS

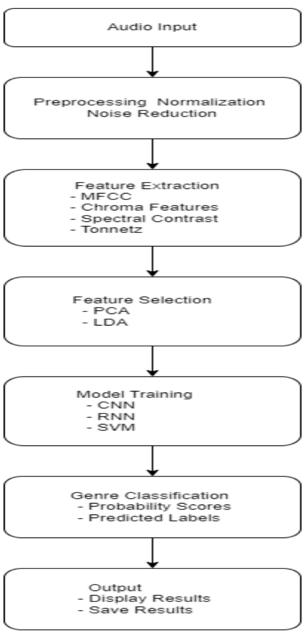
1.TENSOR FLOW

TensorFlow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications. Easily train and deploy models in the cloud, on-perm, in the browser, or on-device no matter what language you use. A simple and flexible architecture to take new ideas from concept to code, to state-of-the-art models, and to publication faster.

TensorFlow allows developers to create data flow graphs—structures that describe how data moves through a graph, or a series of processing nodes. Each node in the graph represents a mathematical operation, and each connection or edge between nodes is a multidimensional data array, or tensor.

III.IMPLEMENTAT IONAND WORKING

1. BLOCK DIAGRAM



1.WORKING

The data set is created consisting several types of audio tones. This data set is used for two purposes training and testing. The deep learning model is trained using data from training data set. The value of epoch is given to the model while training which indicates the number of times a model is trained. Then we test the model with a sample from testing data set. The model gives us what type of music genre it is from theGiven audio tone and it also tells how accurate the model is.

It works as the songs in the data set belongs to different types of categories. It takes audio as an input and verifies the audio signal and it classifies what type of genre that this music audio belongs to.

SOURCE CODE:

```
import os
import librosa
import matplotlib.pyplot as plt
import tensorflow as tf
import numpy as np
from tensorflow.keras.layers import Input, Conv2D, MaxPool2D, Flatten, Dense,Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical
from tensorflow.image import resize
import seaborn as sns

example_file="Data/genres_original/blues/blues.00000.wav" #Copy path
# sr-Sampling Rate
# y,sr=librosa.load(example_file) #Takes default sr
x,sr=librosa.load(example_file,sr=44100)

x.shape
# sr
 (1323588,)

plt.figure(figsize=(14,5))
librosa.display.waveshow(x, sr=sr)

from IPython.display import Audio
Audio(data=x,rate=sr)
audio_path = "./blues.0000.wav"
y,sr= librosa.load(example_file,sr=None)
#sr=None-- keep original sampling Rate
chunk_duration = 4
overlap_duration = 2

#Calculate y for Chunk -- convert duration to Sample
chunk_samples = chunk_duration * sr
overlap_samples = overlap_duration * sr

# Calculate no of chunks
no_of_chunks= int(np.ceil((len(y)-chunk_samples)/(chunk_samples-overlap_samples)))+1

# Iteration of each chunk
for i in range(no_of_chunks):
  start= i* (chunk_samples - overlap_samples) # 0-4, 2-6,
  end = start + chunk_samples
  chunk= y[start:end]
  plt.figure(figsize=(4,2))
  librosa.display.waveshow(chunk,sr=sr)
  plt.show()

def plot_melspectrogram(y,sr):
    # Compute the spectrogram
    spectrogram = librosa.feature.melspectrogram(y=y, sr=sr)
    # Convert to decibels (log scale)
    spectrogram_db = librosa.power_to_db(spectrogram, ref=np.max)
    # Visualize the spectrogram
    plt.figure(figsize=(10, 4))
    librosa.display.specshow(spectrogram_db, sr=sr, x_axis='time', y_axis='mel')
    plt.colorbar(format='%+2.0f dB')
    plt.title('Spectrogram')
    plt.tight_layout()
    plt.show()

def plot_melspectrogram_chunks(y,sr):
    # Define the duration of each chunk and overlap
    chunk_duration = 4  # seconds
    overlap_duration = 2  # seconds

    # Convert durations to samples
    chunk_samples = chunk_duration * sr
    overlap_samples = overlap_duration * sr

    # Calculate the number of chunks
    num_chunks = int(np.ceil((len(y) - chunk_samples) / (chunk_samples - overlap_samples))) + 1

    # Iterate over each chunk
    for i in range(num_chunks):
        # Calculate start and end indices of the chunk
        start = i * (chunk_samples - overlap_samples)
        end = start + chunk_samples

        # Extract the chunk of audio
        chunk = y[start:end]

        # Compute the Mel spectrogram for the chunk
        mel_spectrogram = librosa.feature.melspectrogram(y=chunk, sr=sr)
        print(mel_spectrogram.shape)
        spectrogram_db = librosa.power_to_db(mel_spectrogram, ref=np.max)
        # Visualize the spectrogram
        plt.figure(figsize=(10, 4))
        librosa.display.specshow(spectrogram_db, sr=sr, x_axis='time', y_axis='mel')
        plt.colorbar(format='%+2.0f dB')
        plt.title('Spectrogram')
        plt.tight_layout()
# Define your folder structure
data_dir = 'Data/genres_original'
classes = ['blues', 'classical','country','disco','hiphop','metal','pop','reggae','rock']
```

```
# Load and preprocess audio data
def load_and_preprocess_data(data_dir, classes,
target_shape=(150, 150)):
    data = []
    labels = []

    for i_class, class_name in enumerate(classes):
        class_dir = os.path.join(data_dir, class_name)
        print("Processing--",class_name)
        for filename in os.listdir(class_dir):
            if filename.endswith('.wav'):
                file_path = os.path.join(class_dir,
filename)
                audio_data, sample_rate =
librosa.load(file_path, sr=None)
                # Perform preprocessing (e.g., convert to
Mel spectrogram and resize)
                # Define the duration of each chunk and
overlap
                chunk_duration = 4  # seconds
                overlap_duration = 2  # seconds

                # Convert durations to samples
                chunk_samples = chunk_duration *
sample_rate
                overlap_samples = overlap_duration *
sample_rate

                # Calculate the number of chunks
                num_chunks =
int(np.ceil((len(audio_data) - chunk_samples) /
(chunk_samples - overlap_samples))) + 1

                # Iterate over each chunk
                for i in range(num_chunks):
                    # Calculate start and end indices of
the chunk
                    start = i * (chunk_samples -
overlap_samples)
                    end = start + chunk_samples

                    # Extract the chunk of audio
                    chunk = audio_data[start:end]

                    # Compute the Mel spectrogram for
the chunk
                    mel_spectrogram =
librosa.feature.melspectrogram(y=chunk, sr=sr)

                #mel_spectrogram =
librosa.feature.melspectrogram(y=audio_data,
sr=sample_rate)
                    mel_spectrogram =
resize(np.expand_dims(mel_spectrogram, axis=-1),
target_shape)
                    data.append(mel_spectrogram)
                    labels.append(i_class)
    return np.array(data), np.array(labels)
# Split data into training and testing sets
data, labels = load_and_preprocess_data(data_dir,
classes)
#print("\nData:",data,"\nlabel",labels)
Processing-- blues
Processing-- classical
Processing-- country
Processing-- disco
Processing-- hiphop
Processing-- metal
Processing-- pop
Processing-- reggae
Processing-- rock
data.shape
(13490, 150, 150, 1)
labels.shape
(13490,)
labels
array([0, 0, 0, ..., 8, 8, 8])
labels = to_categorical(labels,
num_classes=len(classes))  # Convert labels to one-
hot encoding
labels
array([[1., 0., 0., ..., 0., 0., 0.],
       [1., 0., 0., ..., 0., 0., 0.],
       [1., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 1.],
       [0., 0., 0., ..., 0., 0., 1.],
       [0., 0., 0., ..., 0., 0., 1.]])

labels.shape
(13490, 9)
data.shape
(13490, 150, 150, 1)
from sklearn.model_selection import
train_test_split
X_train, X_test, y_train, y_test =
train_test_split(data, labels, test_size=0.2,
random_state=42)
#model = tf.keras.models.Sequential()
X_train[0].shape


(150, 150, 1)
input_shape=(150, 150, 1)
model = tf.keras.models.Sequential()
model.add(Conv2D(filters=32,kernel_size=3,paddin
g='same',activation='relu',input_shape=X_train[0].s
hape))
model.add(Conv2D(filters=32,kernel_size=3,activat
ion='relu'))
model.add(MaxPool2D(pool_size=2,strides=2))
model.add(Conv2D(filters=64,kernel_size=3,paddin
g='same',activation='relu'))
```

```
model.add(Conv2D(filters=64,kernel_size=3,activation='relu'))
model.add(MaxPool2D(pool_size=2,strides=2))
model.add(Conv2D(filters=128,kernel_size=3,padding='same',activation='relu'))
model.add(Conv2D(filters=128,kernel_size=3,activation='relu'))
model.add(MaxPool2D(pool_size=2,strides=2))
model.add(tf.keras.layers.Dropout(0.3))
model.add(Conv2D(filters=256,kernel_size=3,padding='same',activation='relu'))
model.add(Conv2D(filters=256,kernel_size=3,activation='relu'))
model.add(MaxPool2D(pool_size=2,strides=2))
model.add(Conv2D(filters=512,kernel_size=3,padding='same',activation='relu'))
model.add(Conv2D(filters=512,kernel_size=3,activation='relu'))
model.add(MaxPool2D(pool_size=2,strides=2))
model.add(Dropout(0.3))
model.add(Flatten())
model.add(Dense(units=1200,activation='relu'))
model.add(Dropout(0.45))
model.add(Dense(units=len(classes),activation='softmax'))
model.summary()


model.compile(optimizer=Adam(learning_rate=0.0001), loss='categorical_crossentropy', metrics=['accuracy'])
X_train.shape,y_train.shape
((10792, 150, 150, 1), (10792, 9))
training_history = model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(X_test, y_test))
Epoch 1/10
338/338 ━━━━━━━━━━━━━━━━━━━━ 2442s 7s/step - accuracy: 0.2252 - loss: 2.0303 - val_accuracy: 0.4211 - val_loss: 1.5778
Epoch 2/10
338/338 ━━━━━━━━━━━━━━━━━━━━ 2148s 6s/step - accuracy: 0.4819 - loss: 1.4469 - val_accuracy: 0.6012 - val_loss: 1.1371
Epoch 3/10
338/338 ━━━━━━━━━━━━━━━━━━━━ 1147s 3s/step - accuracy: 0.6074 - loss: 1.1256 - val_accuracy: 0.6312 - val_loss: 1.0535
Epoch 4/10
338/338 ━━━━━━━━━━━━━━━━━━━━ 1745s 5s/step - accuracy: 0.6839 - loss: 0.9168 - val_accuracy: 0.7368 - val_loss: 0.7908
Epoch 5/10
338/338 ━━━━━━━━━━━━━━━━━━━━ 2172s 6s/step - accuracy: 0.7253 - loss: 0.8005 - val_accuracy: 0.7113 - val_loss: 0.8585
Epoch 6/10
338/338 ━━━━━━━━━━━━━━━━━━━━ 2219s 6s/step - accuracy: 0.7706 - loss: 0.6711 - val_accuracy: 0.7880 - val_loss: 0.6331
Epoch 7/10
338/338 ━━━━━━━━━━━━━━━━━━━━ 6799s 20s/step - accuracy: 0.7933 - loss: 0.5973 - val_accuracy: 0.7891 - val_loss: 0.6064
Epoch 8/10
338/338 ━━━━━━━━━━━━━━━━━━━━ 657s 2s/step - accuracy: 0.8316 - loss: 0.5035 - val_accuracy: 0.8002 - val_loss: 0.5793
Epoch 9/10
338/338 ━━━━━━━━━━━━━━━━━━━━ 684s 2s/step - accuracy: 0.8558 - loss: 0.4282 - val_accuracy: 0.8202 - val_loss: 0.5166
Epoch 10/10
338/338 ━━━━━━━━━━━━━━━━━━━━ 745s 2s/step - accuracy: 0.8688 - loss: 0.3784 - val_accuracy: 0.8462 - val_loss: 0.4464
model.save("Trained_model.h5") #Windows


import tensorflow as tf
import numpy as np
import librosa
def preprocess_file(file_path):
  target_shape=(150, 150)
  data = []
  audio_data, sample_rate = librosa.load(file_path, sr=None)

  chunk_duration = 4  # seconds
  overlap_duration = 2  # seconds

           # Convert durations to samples
  chunk_samples = chunk_duration * sample_rate
  overlap_samples = overlap_duration * sample_rate

 # Calculate the number of chunks
  num_chunks = int(np.ceil((len(audio_data) - chunk_samples) / (chunk_samples - overlap_samples))) + 1

# Iterate over each chunk
  for i in range(num_chunks):

             # Calculate start and end indices of the chunk
     start = i * (chunk_samples - overlap_samples)
     end = start + chunk_samples

             # Extract the chunk of audio
     chunk = audio_data[start:end]
   # Compute the Mel spectrogram for the chunk
```

```
        mel_spectrogram =
librosa.feature.melspectrogram(y=chunk,
sr=sample_rate)
    #mel_spectrogram =
librosa.feature.melspectrogram(y=audio_data,
sr=sample_rate)
        mel_spectrogram =
np.resize(np.expand_dims(mel_spectrogram,
axis=-1), target_shape)
        data.append(mel_spectrogram)
        #labels.append(i_class)

    return np.array(data)


file_path =
r'Data\genres_original\country\country.00009.wav'
test_data=preprocess_file(file_path)
model =
tf.keras.models.load_model(r'Trained_model.
keras' predictions =model.predict(test_data)
1/1 ━━━━━━━━━━━━━━━━━━━━
━━ 1s 960ms/step
genre_index = np.argmax(predictions, axis=1)
genres=['blues',
'classical','country','disco','hiphop','metal','pop','reg
gae','rock']
predicted_genre = genres[genre_index[0]]
print(predicted_genre)
)
```

Results:

The given audio is successfully classified by the deeplearning model, depending on the input audio given.

```
In [23]: file_path = r'Data\genres_original\country\country.00009.wav'
         test_data=preprocess_file(file_path)
```

```
In [25]: model = tf.keras.models.load_model(r'Trained_model.keras')
```

```
In [28]: predictions =model.predict(test_data)

         1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 960ms/step
```

```
In [32]: genre_index = np.argmax(predictions, axis=1)
```

```
In [34]: genres=['blues', 'classical','country','disco','hiphop','metal','pop','reggae','rock']
         predicted_genre = genres[genre_index[0]]
         print(predicted_genre)

         country
```

Future enhancements:

This model can be further implemented using various musical datasets. There is an also scope for Combining multiple models to create more robust classifiers more efficient and useful. The accuracy of the model can be further increasedby using more data set. Developing more sophisticated audio feature extraction techniques to capture nuanced aspects of music

Conclusion:

The deep learning model is successfully built and trained for classifying the music genre from the data.