Project Description

GROUP MEMBERS:
Sanika Chandode - 201071045
Chaitali Chaudhari - 201071060
Grishma Barule - 201071062
Dhanashree Pawar - 201071061

AIM: Music genre classification using Neural Networks and KNN.

THEORY:

Music genre classification is the task of automatically assigning a genre label to a piece of music based on its acoustic features. This task is important in various applications such as music recommendation, music search, and playlist generation. However, music genre classification is a challenging problem because of the high variability and subjective nature of music genres.

One of the popular approaches to music genre classification is to use machine learning techniques such as neural networks and k-NN. Neural networks are powerful models that can learn complex non-linear relationships between the input features and output labels. They have been successfully applied to various music classification tasks, including music genre classification. On the other hand, k-NN is a simple and effective method that can classify a new instance based on the majority vote of its k-nearest neighbors in the training set.

DATASET:

The dataset used for this implementation is the GTZAN dataset, which consists of 100 samples for each of the 10 music genres. The genres include blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, and rock.

- **Genres original** - A collection of 10 genres with 100 audio files each, all having a length of 30 seconds (the famous GTZAN dataset, the MNIST of sounds)
- **images original** - A visual representation for each audio file. One way to classify data is through neural networks. Because NNs (like CNN, what we will be using today) usually take in some sort of image representation, the audio files were converted to Mel Spectrograms to make this possible.
- **2 CSV files** - Containing features of the audio files. One file has for each song (30 seconds long) a mean and variance computed over multiple features that can be extracted from an audio file. The other file has the same structure, but the songs were split before into 3 seconds audio files (this way increasing 10 times the amount of data we fuel into our classification models). *With data, more is always better*.

MFCC Features:

MFCC (Mel-Frequency Cepstral Coefficients) features are commonly used in audio signal processing and analysis. They are a type of spectral feature that captures the shape of the short-term power spectrum of a sound signal, based on the mel scale of human hearing.

MFCCs are often used for speech recognition and music genre classification tasks because they provide a compact representation of the spectral characteristics of an audio signal, which can be used to extract relevant features and patterns for machine learning algorithms. The computation of MFCCs involves several steps, including pre-emphasis, framing, windowing, Fourier transform, mel-frequency wrapping, and cepstral analysis. The resulting MFCC feature vector typically consists of a set of coefficients that capture the frequency and amplitude information of the audio signal in a compact and informative way.

Using Neural Networks:

The music genre classification project involves building a neural network model that can classify music files into one of ten different genres. The genres include blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, and rock. The project uses the Librosa library to extract MFCC features from the audio files and trains a deep learning model using the Keras library. The model consists of four layers, with the first layer having 128 neurons, the second layer having 64 neurons, the third layer having 32 neurons, and the final layer having ten neurons to represent the ten different genres.

The dataset used in the project consists of 100 audio files for each genre, with each file having a length of 30 seconds. The project reads the first 100 files for each genre and extracts MFCC features from each file. It then stores the features and corresponding labels in arrays and uses them to train and evaluate the model. The LabelEncoder class is used to encode the genre labels as integers.

Once the model is trained and saved, it can be used to classify new audio files into one of the ten genres. To classify a new file, the model first extracts the MFCC features from the file and then uses the predict() method to obtain the probabilities of the file belonging to each genre. The genre with the highest probability is then considered as the predicted genre for the file.

Using KNN:

K-Nearest Neighbors Algorithm. The k-nearest neighbors algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point.

We select the number K of the neighbors.Calculate the Euclidean distance of K number of neighbors.Take the K nearest neighbors as per the calculated Euclidean distance. Among these k neighbors, count the number of the data points in each category.Assign the new data points to that category for which the number of the neighbor is maximum.

Here we first load the GTZAN dataset and then extract features from each audio file using the librosa library, which is a Python package for music and audio analysis. The extracted features are the Mel-frequency cepstral coefficients (MFCCs) of each audio file. MFCCs are commonly used for audio processing and represent the spectral envelope of a signal. The MFCCs are then averaged across the signal to produce a single feature vector for each audio file.
The KNN algorithm is then trained on the extracted features and the corresponding music genres. The implementation uses the scikit-learn library, which is a machine learning library for Python. The KNN classifier calculates the distance between instances and trained with k value of 5 hence, finds the 5 nearest neighbors.
Finally, the trained KNN classifier is used to predict the genre of new audio files. To do this, the MFCCs of the new audio file are extracted using librosa, and the KNN classifier is used to predict the genre. The predicted genre is then returned.

OUTPUT:

1) **Using Neural Network:**

Accuracy:

```
# Evaluating the Model on the Testing Data
loss, accuracy = model.evaluate(X_test, y_test)
print("Accuracy:", accuracy)
```

```
25/25 [==============================] - 0s 2ms/step - loss: 0.0238 - accuracy: 0.9962 - val_loss: 2.5095 - val_accuracy: 0.5
850
Epoch 96/100
25/25 [==============================] - 0s 2ms/step - loss: 0.0268 - accuracy: 0.9962 - val_loss: 2.6382 - val_accuracy: 0.5
550
Epoch 97/100
25/25 [==============================] - 0s 2ms/step - loss: 0.0233 - accuracy: 0.9975 - val_loss: 2.4637 - val_accuracy: 0.6
100
Epoch 98/100
25/25 [==============================] - 0s 2ms/step - loss: 0.0270 - accuracy: 0.9962 - val_loss: 2.5189 - val_accuracy: 0.6
050
Epoch 99/100
25/25 [==============================] - 0s 2ms/step - loss: 0.0294 - accuracy: 0.9962 - val_loss: 2.6232 - val_accuracy: 0.5
800
Epoch 100/100
25/25 [==============================] - 0s 2ms/step - loss: 0.0309 - accuracy: 0.9962 - val_loss: 2.7489 - val_accuracy: 0.5
750
7/7 [==============================] - 0s 2ms/step - loss: 2.7489 - accuracy: 0.5750
Accuracy: 0.574999988079071
```

Predictions:

```
# Save the trained model
model.save('my_model.h5')

# Load the saved model
loaded_model = keras.models.load_model('my_model.h5')

# Predict the genre of an audio file
mfccs = extract_feature('C:/Users/ADMIN/Downloads/Data/audio_file.wav')
input_data = mfccs.reshape(1, -1)
predicted_genre = loaded_model.predict(input_data)

# Extract genre name from probabilities
genres = ["blues", "classical", "country", "disco", "hiphop", "jazz", "metal", "pop", "reggae", "rock"]
predicted_genre_index = np.argmax(predicted_genre)
predicted_genre_name = genres[predicted_genre_index]
print('Predicted genre: ', predicted_genre_name)
```

```
1/1 [==============================] - 0s 63ms/step
Predicted genre:  rock
```

2) Using KNN:

Accuracy:

```
y_pred = knn.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
```

```
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy
(detected version 1.23.5
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
Processing genre: blues
Processing genre: classical
Processing genre: country
Processing genre: disco
Processing genre: hiphop
Processing genre: jazz
/tmp/ipykernel_34/421666.py:24: UserWarning: PySoundFile failed. Trying audioread instead.
  audio_data, sr = librosa.load(file_path, res_type="kaiser_fast")
/opt/conda/lib/python3.10/site-packages/librosa/core/audio.py:184: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
Error processing /kaggle/input/gtzan-dataset-music-genre-classification/Data/genres_original/jazz/jazz.00054.wav:
Processing genre: metal
Processing genre: pop
Processing genre: reggae
Processing genre: rock
Accuracy: 0.5
```

Predictions:

```
# Extract MFCC features
audio_features = librosa.feature.mfcc(y=audio_data, sr=sr, n_mfcc=20)
audio_features = np.mean(audio_features, axis=1)
audio_features = audio_features.reshape(1, -1)

# Make prediction using KNN classifier
genre_mapping = {0: "blues", 1: "classical", 2: "country", 3: "disco", 4: "hiphop",
                 5: "jazz", 6: "metal", 7: "pop", 8: "reggae", 9: "rock"}
knn_pred = knn.predict(audio_features)
predicted_genre = genre_mapping[knn_pred[0]]

print(f"Predicted genre: {predicted_genre}")
```

```
Predicted genre: rock
```

+ Code + Markdown

CONCLUSION:

From the two models that we trained, for Music Genre Classification,we find that Neural Networks have a higher rate of accuracy than KNN. Therefore, Neural Network for Music Genre Classification is determined to be a better model than KNN.