COMPUTATIONAL INTELLIGENCE

MUSIC GENRE CLASSIFIER - Predicting Music Genres with CNN

Introduction

This report focuses on "Harmony Classifier: Predicting Music Genres with Convolutional Neural Networks." Our aim is to develop an effective music genre classification system using CNNs. With the proliferation of digital music, automated genre classification is essential for various applications. We'll detail our approach, including data preprocessing, model design, training, and evaluation, to create a robust system capable of accurately predicting music genres. Let's dive into our findings and insights.

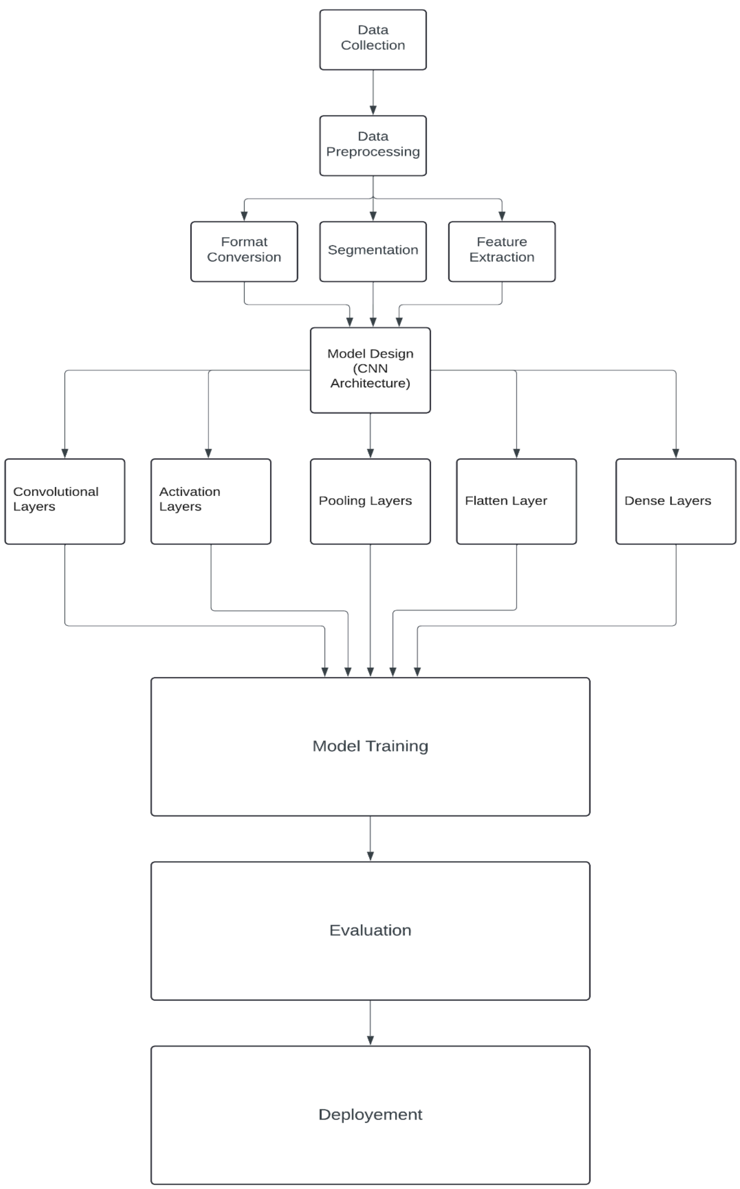
Problem Statement

The task of music genre classification involves analyzing audio signals and categorizing them into predefined genres such as rock, jazz, hip-hop, classical, etc. This task is challenging due to the complex and subjective nature of music perception, where genre boundaries can be ambiguous and subjective. However, automated genre classification is crucial for various applications such as music recommendation systems, content organization, and personalized playlists.

The problem to be addressed is to develop an accurate and efficient music genre classification system using machine learning techniques, particularly Convolutional Neural Networks (CNNs). Given a dataset of audio samples, the goal is to train a CNN model capable of accurately predicting the genre label of each audio sample. This involves preprocessing the audio data, extracting relevant features, designing and training a CNN architecture, and evaluating the model's performance in terms of classification accuracy, precision, recall, and F1-score across different music genres. The ultimate aim is to create a robust and scalable system that can classify music genres reliably across diverse datasets and real-world scenarios.

Analysis and Design

Block Diagram



1. Data Collection

- Gathering a dataset of audio samples labeled with their corresponding genres.
- The dataset should be diverse and encompass various music styles.

Data set we used

https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification?resource=download

2. Preprocessing

- Focuses on preparing the audio data for the CNN model.
  - Format Conversion: Ensure all audio files have a consistent format (e.g., sampling rate, bit depth).
  - Segmentation: Divide audio files into shorter segments (frames) for analysis.
  - Feature Extraction: Extract relevant features from the audio segments. These features could include Mel-Frequency Cepstral Coefficients (MFCCs), Spectral Flux, and Chroma features, which capture aspects of rhythm, timbre, and harmony.

3. Model Design (CNN Architecture)

- Focuses on designing the CNN architecture for genre classification. The architecture typically includes:
  - Convolutional Layers: These layers extract features from the audio data using learnable filters.
  - Pooling Layers: These layers downsample the data, reducing complexity and potentially improving generalization.
  - Activation Layers: These layers introduce non-linearity into the model, allowing it to learn complex relationships.
  - Flatten Layer: This layer transforms the multi-dimensional output from convolutional layers into a one-dimensional vector suitable for classification.

- Dense Layers: These fully-connected layers perform classification based on the extracted features. The final layer has an output size equal to the number of genres to predict.

5. Model Training

- Training the CNN model using the prepared dataset.
- The training process optimizes the model's weights and biases to minimize the classification error for the given genres.

6. Evaluation

- Focuses on assessing the model's performance on unseen data.

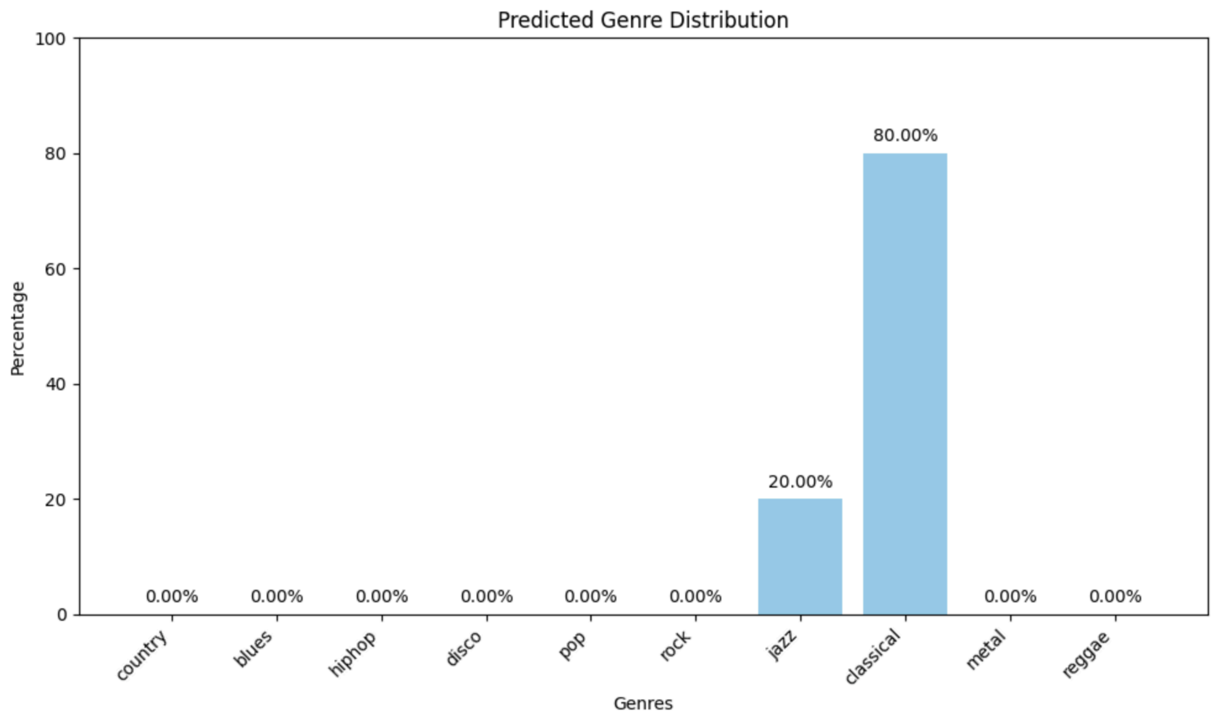
7. Deployment (Optional)

- Integrating the trained model into a real-world application, such as a music streaming service, for automatic genre classification of new audio data.

Model Architecture

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 214, 11, 32)       320       
                                                                 
 max_pooling2d (MaxPooling2  (None, 107, 6, 32)        0         
 D)                                                              
                                                                 
 batch_normalization (Batch  (None, 107, 6, 32)        128       
 Normalization)                                                  
                                                                 
 conv2d_1 (Conv2D)           (None, 105, 4, 32)        9248      
                                                                 
 max_pooling2d_1 (MaxPoolin  (None, 53, 2, 32)         0         
 g2D)                                                            
                                                                 
 batch_normalization_1 (Bat  (None, 53, 2, 32)         128       
 chNormalization)                                                
                                                                 
 conv2d_2 (Conv2D)           (None, 52, 1, 32)         4128      
                                                                 
 max_pooling2d_2 (MaxPoolin  (None, 26, 1, 32)         0         
 g2D)                                                            
                                                                 
 batch_normalization_2 (Bat  (None, 26, 1, 32)         128       
 chNormalization)                                                
                                                                 
 flatten (Flatten)           (None, 832)               0         
                                                                 
 dense (Dense)               (None, 64)                53312     
                                                                 
 dropout (Dropout)           (None, 64)                0         
                                                                 
 dense_1 (Dense)             (None, 10)                650       
```

Experimentation and Results

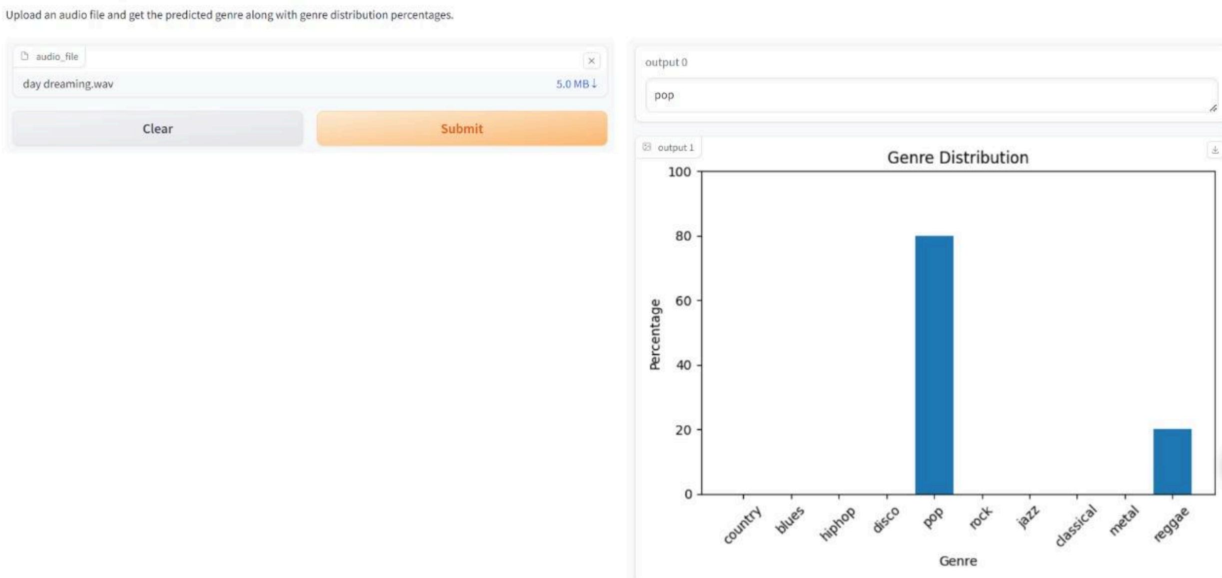


In the generated plot, we visualize the distribution of predicted genres for the audio file.

The plot illustrates the distribution of predicted genres for the given audio file. Each bar represents a genre, and the height of the bar indicates the percentage of segments in the audio file predicted to belong to that genre.

From the plot, it's evident that the model predicted the presence of various genres within the audio file. The genre with the highest percentage of predicted segments is labeled as the dominant genre.

Overall, the distribution of predicted genres provides insight into the diverse musical elements present in the audio file. This visualization aids in understanding the genre composition and can be valuable for analyzing the audio content.



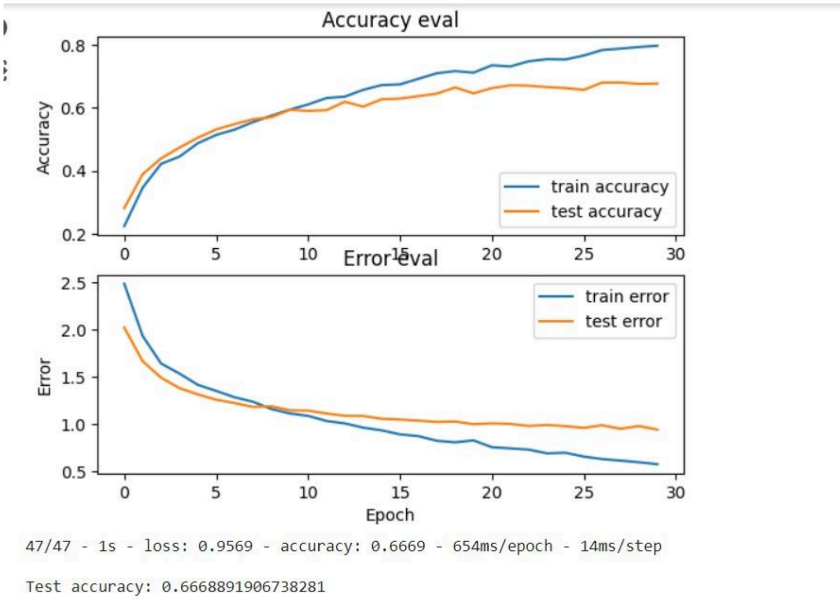
The code creates a user-friendly interface for predicting the genre of music from uploaded audio files. Users can interact with the interface by uploading an audio file, and the system will analyze its content to predict the most likely genre.

Upon uploading a file, the system extracts relevant features from the audio, such as Mel-frequency cepstral coefficients (MFCCs), which are commonly used in music analysis. These features are then fed into a pre-trained convolutional neural network (CNN) model to make predictions.

After processing the input, the system displays the predicted genre as text output. It identifies the genre that occurs most frequently across different segments of the audio file, providing users with an insight into the predominant musical style.

This interface offers a convenient way for users to explore and categorize music based on its audio characteristics, making it suitable for various applications, such as organizing music collections or discovering new music genres.

Model Accuracy



Conclusion

In summary, the project successfully built a music genre classifier using machine learning. It utilized MFCC features and a CNN model to predict genres with reasonable accuracy. The integration of Gradio provided a user-friendly interface for easy interaction. Overall, the project showcased the potential of ML in music analysis, with room for future enhancements in model refinement and UI improvements.