# **Music Genre Classification Project Report**

## **Project Overview**

This project creates a computer program that can listen to music and guess what type of music it is. The program can tell the difference between 10 different music styles: blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae, and rock. It works by looking at patterns in the sound and learning what makes each type of music different.

## **Data Types and Structures Used**

The project uses several different ways to store numbers. **Decimal numbers** are used to store the actual sound information from music files. When the computer reads a music file, it turns the sound into decimal numbers that represent the audio waves.

**Whole numbers** are used to represent the music genres. Instead of using words like "rock" or "jazz", the program gives each genre a number from 0 to 9. This makes it easier for the computer to work with the genre labels.

**Tensors** are from PyTorch used for machine learning. The program converts regular number lists into these tensors so the neural network can process them efficiently.

**Lists** are used throughout the project to keep related information together. The audio\_files list holds all the file names, the labels list holds genre numbers, and the features\_list accumulates processed audio data. These lists maintain order, so each song matches with its correct genre.

**Arrays** organize the processed audio information in neat rows and columns, creating grids that represent different frequencies and time periods in the music.

## **Data Processing and Transformation**

The program performs several important data preparation steps. First, it standardizes all audio files to exactly 30 seconds using silence ending for short files and trimming for long ones. This ensures consistent input sizes for the neural network.

The audio processing extracts mel-spectrograms, which convert sound waves into visual patterns showing how different frequencies change over time. These spectrograms undergo normalization using z-score standardization, where the program calculates the average and standard deviation of all values, then adjusts them to have a standard range. This helps the neural network learn more effectively.

The program splits data randomly into 80% for training and 20% for testing. This random splitting prevents bias and ensures the model evaluation is fair. The max\_files parameter allows limiting dataset size during development, making testing faster while maintaining the ability to use full datasets later.

## **Object-Oriented Programming Structure**

The main part of the program uses a **class** called MusicCNN that inherits from PyTorch's neural network class. This inheritance automatically provides machine learning capabilities without writing extra code.

The program uses **functions** for different tasks: loading music files, processing audio, training the network, and making predictions. Each function has a specific job, making the code reusable and easier to debug. If there's a problem with one part, you only need to fix that specific function.

## **Model Training and Architecture**

The neural network uses a convolutional architecture with three main blocks. Each block contains pattern detection layers, activation functions, and pooling layers that reduce data size while keeping important information. The final classification section uses fully connected layers to make genre predictions.

Training uses the Adam optimizer with a learning rate of 0.001 and CrossEntropyLoss for calculating errors. The model trains for 10 epochs, with loss values printed every 2 epochs to monitor progress. During each epoch, the program calculates prediction errors, adjusts the network weights, and improves accuracy.

The training process uses the CPU due to the pc build i have excels with data processing

## **Model Evaluation and Performance**

The model evaluation uses accuracy as the primary metric, calculating the percentage of correct predictions on the test dataset. After training, the program evaluates performance using the held-out test data that wasn't seen during training.

The evaluation process converts test data to tensors, runs predictions through the trained model, and compares predicted genres with actual labels. The accuracy calculation shows how well the model generalizes to new, unseen music files.

The prediction function also provides confidence scores using softmax, showing how certain the model is about each prediction. This gives users insight into prediction reliability.

## **Hyperparameter Choices and Optimization**

The project uses several fixed hyperparameters based on common practices. The learning rate of 0.001 provides stable training without being too slow or too fast. The 10-epoch training period balances learning time with computational efficiency. This can however be adjusted to increase improvements but will require more time due to more epoches

The mel-spectrogram parameters include 128 mel-frequency bands and specific window sizes that capture important audio characteristics. The 30-second audio length ensures sufficient information for genre classification while maintaining reasonable processing times.

The neural network architecture uses increasing filter sizes (32, 64, 128) to detect patterns at different complexity levels, from simple frequency patterns to complex musical characteristics.

## **Key Insights and Results**

The project demonstrates that convolutional neural networks can effectively classify music genres by analysing audio spectrograms. The model learns to identify distinctive patterns that characterize different musical styles.

The modular code structure proves beneficial for development and maintenance. Separating data processing, model definition, training, and evaluation into distinct functions makes the system easier to understand and modify.

The error handling approach, where corrupted files return default values instead of crashing the program, shows the importance of robust data processing in real-world applications.

## **Conclusion**

This music genre classification project successfully combines appropriate data types, object-oriented programming, and systematic data processing to solve a practical problem. The model training process demonstrates effective use of neural networks for audio classification, while the evaluation shows the system's ability to generalize to new music. The modular design supports future improvements and extensions to additional genres or audio analysis tasks. For the future, i will implement functions where you can change the max files and epoch values to make this code training more effective without needing to go inside the code to change theses.