Music Genre Classification

Ouyang Zhuowen
1023040926)
Nanjing University of Posts and Telecommunications
Jiangsu, Nanjing China

***Abstract*—This paper aims to train deep learning models to predict the genre of a 30- second song clip. The study designs two models for music genre classification: a transfer learning model based on the InceptionV3 model and a custom convolutional neural network (CNN) model. By comparing the advantages, disadvantages, and final performance of these two models, the paper evaluates their performance in the task of music genre classification.The content of the paper is divided into two main parts. The first part focuses on data preprocessing and visualization of the audio. In this section, the paper explains how to process the raw dataset, including loading the audio files and extracting features. Additionally, it discusses how to visualize the audio data, such as plotting waveforms, spectrograms, or other relevant audio feature visualizations.The second part covers the training and evaluation of the models required for this study. It provides a detailed description of the architectures, parameter settings, and training processes of the two models. Finally, the performance of these models is evaluated and compared, possibly using metrics such as accuracy to measure the classification performance. The paper also discusses the strengths and limitations of the models.**

***Index Terms*—Terms—music genre classification , Transfer Learning Model, convolutional neural networks**

I. Introduction

the explosive growth of music content and the popularity of digital music platforms, there is an increasing demand for automated music genre classification and recognition. Music genre classification is an important research task that aims to automatically categorize music tracks into different genres or categories through machine learning modeling. It plays a significant role in applications such as music recommendation systems, personalized playlists, and music retrieval.

However, music genre classification faces several challenges. Firstly, music genres are subjective concepts, and different individuals may have differences in classifying the same song into genres. Secondly, music genre features are diverse and complex, including aspects such as timbre, rhythm, and harmony. Extracting these features and effectively representing them is a challenging task.

In recent years, the development of machine learning has brought new opportunities to music genre classification. In particular, the rise of deep learning technology allows deep neural networks to learn higher-level abstract features from audio signals, thereby enhancing the performance of music genre classification. Deep learning models can automatically learn the temporal and spectral features of music and capture the differences between different genres.

By reviewing and analyzing music genre classification methods, we can gain in-depth understanding of the strengths and limitations of different approaches and explore future research directions. The progress in music genre classification will contribute to improving the personalized capabilities of music recommendation systems, providing better music experiences, and promoting the dissemination and development of music culture.

II. Related Work

Currently, Python has a built-in library called librosa, which is the primary tool for handling audio files. With librosa, you can perform operations such as audio feature extraction and visualization. However, to achieve your ultimate goal, relying solely on librosa might not be sufficient. Here are the main libraries commonly used in conjunction with librosa:

1)librosa: Librosa is a powerful Python library for audio analysis. It is suitable for extracting features, analyzing audio content, and processing audio data. Librosa provides rich functionality, including audio feature extraction, audio visualization, beat and rhythm analysis, audio transformation, and processing. Integrated with libraries like NumPy and SciPy, Librosa simplifies audio analysis and makes it efficient.

2)Matplotlib: Matplotlib is a plotting and data visualization library for the Python programming language. It is suitable for creating simple plots to complex visualizations. With its extensive features and flexible interface, Matplotlib allows users to effortlessly generate high-quality charts for data presentation and analysis. It can be combined with Python libraries like NumPy and SciPy for scientific computing.

3)pandas: Pandas is a powerful Python library for data processing and analysis. It provides simple and efficient data structures and tools. The core of pandas consists of Series and DataFrame, capable of handling various data types. Pandas supports tasks such as data import/export, data cleaning and preprocessing, data selection and filtering, data grouping and aggregation, time series analysis, and data visualization.

III. Data Preprocessing

In this experiment, the dataset chosen is the GTZAN dataset, which is one of the most famous music classification datasets. It was used in the paper "Musical genre classification of audio signals" by G. Tzanetakis and P. Cook in 2002. The dataset consists of 1000 audio files recorded from CDs and other sources, with a total of 10 music genres: blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, and rock. Each music genre has 100 files, and the files are in the WAV format.

To make the neural network suitable for audio data, a common approach is to convert the audio into a visual representation (such as a spectrogram) before feeding it into an image classifier. Unlike traditional Hertz spectrograms, the Mel spectrogram better represents human auditory perception. It uses a logarithmic scale instead of a linear scale, emphasizing frequency bands that are more perceptually relevant to human listeners. Below are some visualizations of the audio files:

First, Figure 1 shows the raw waveform graph, which visualizes the amplitude variation of the audio signal over the time axis. It displays how the amplitude of the audio signal changes over time, allowing us to intuitively observe the strength of the sound, the shape of the waveform, and the characteristics of the audio signal. In the raw waveform graph, time is typically represented on the horizontal axis, with units such as seconds or milliseconds, while the amplitude of the audio signal is represented on the vertical axis. The raw waveform graph enables quick scanning of the audio data and visual comparison to identify which genres may be more similar to each other.

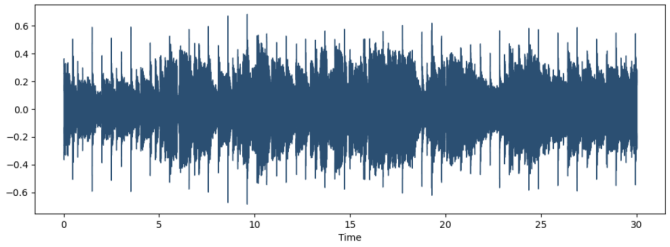


Fig. 1. Raw waveform

Next, Figure 2 shows the spectrogram, which is a visual representation of how the signal's intensity varies with time across different frequencies in a specific waveform. With the spectrogram, we can not only observe whether there is more or less energy at frequencies like 2 Hz and 10 Hz but also see how the energy levels change over time.

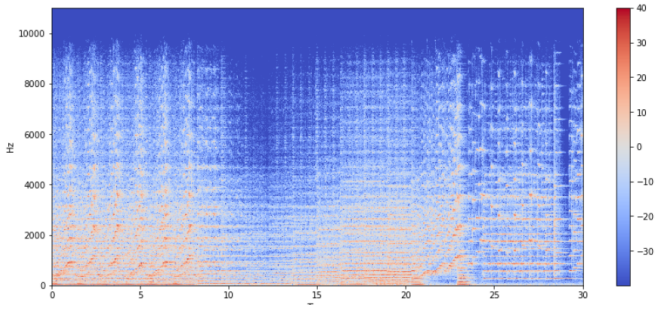


Fig. 2. spectrogram

In Figure 3, the vertical axis of the spectrogram represents frequency (ranging from 0 to 10 kHz), while the horizontal axis represents the time of the clip. As shown in Figure 3, when we convert the frequency axis to a logarithmic scale, we can observe that all the action (highlighted in red) occurs at the bottom of the spectrogram.

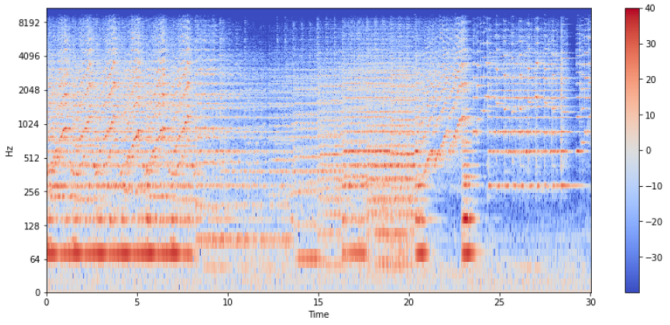


Fig. 3. spectrogram

Figure 4 displays the spectral rolloff, which is a feature commonly used in audio signal processing to describe the decay of the spectral energy in the frequency domain. It represents at which frequency in the spectrum a certain percentage of the total energy is contained. The Spectral Rolloff feature is calculated using the feature.spectral_rolloff function, and the original audio signal is visualized in waveform format. This helps understand the spectral decay of the audio signal and the shape characteristics of the waveform.

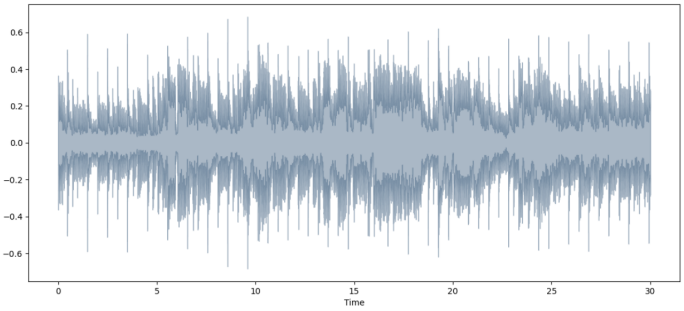


Fig. 4. Spectral Rolloff

Figure 5 displays the chroma feature, which is a representation that categorizes pitch in a meaningful way (typically into twelve categories) and approximates the equal-tempered scale. One of its primary characteristics is capturing the harmonic and melodic features of music while being robust to changes in timbre and instruments. The chroma feature is beneficial for analyzing the tonal content of audio, and it can be utilized in various tasks such as chord recognition, melody extraction, and music genre classification.

After visualizing the audio files, the next step is to preprocess the data from the GTZAN dataset to ensure it is suitable for modeling purposes. Firstly, it is important to check if all the provided files in the dataset are error-free to ensure consistency in the data framework and prevent potential issues in the future.

Figure 6 reveals that there is a missing or corrupted file in the jazz music dataset. Upon further investigation, it appears

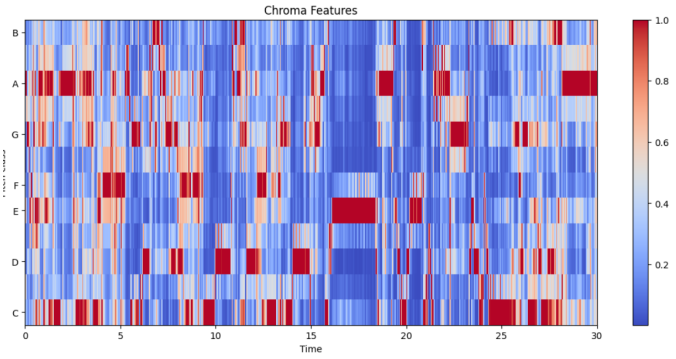


Fig. 5. Chroma Feature

that the file 'jazz.00054.wav' is damaged in the 'genres_yuan' directory, which means no spectrogram has been generated for it in the 'images_yuan' directory.

To overcome this problem and avoid potential biases, it is recommended to reduce all the genre datasets to only 99 files during the training-validation-testing split process. This ensures that each genre has an equal representation and eliminates the influence of the missing or corrupted file on the analysis.

```
Size of blues dataset: 100 files.
Size of classical dataset: 100 files.
Size of country dataset: 100 files.
Size of disco dataset: 100 files.
Size of hiphop dataset: 100 files.
Size of jazz dataset: 99 files.
Size of metal dataset: 100 files.
Size of pop dataset: 100 files.
Size of reggae dataset: 100 files.
Size of rock dataset: 100 files.
```

Fig. 6. File accuracy

Once it is ensured that the data is free of issues, the available data can be divided into three subsets: training, validation, and testing. As shown in Figure 7, the data is split using an 80:9:10 ratio for training, validation, and testing, respectively.

```
Training: {'blues': 80, 'classical': 80, 'country': 80, 'disco': 80, 'hiphop': 80, 'jazz': 8
0, 'metal': 80, 'pop': 80, 'reggae': 80, 'rock': 80}
Validation: {'blues': 9, 'classical': 9, 'country': 9, 'disco': 9, 'hiphop': 9, 'jazz': 9, 'm
etal': 9, 'pop': 9, 'reggae': 9, 'rock': 9}
Test: {'blues': 10, 'classical': 10, 'country': 10, 'disco': 10, 'hiphop': 10, 'jazz': 10, 'm
etal': 10, 'pop': 10, 'reggae': 10, 'rock': 10}
```

Fig. 7. Splitting the Data

IV. MUSIC CLASSIFICATION

After the data has been properly curated and formatted for machine interpretation, the next step is to prepare for the establishment of prediction models. In this experiment, two different models were built to compare and observe which model performs better in predicting music genres. One of the models is the Transfer Learning Model with InceptionV3 Base (CNN). After researching and testing various pre-trained models, the InceptionV3 architecture appeared to be the most effective for this application.

For this experiment, the default ImageNet weights will be used as they have learned to efficiently extract hierarchical features and other abstractions from images. Moreover, ImageNet weights are typically optimized for detecting spatial patterns in images, which will help in detecting frequency and volume patterns over time. At the same time, the weights will be frozen to retain valuable pre-learned features, preventing overfitting, reducing computational costs, and ensuring consistency. This approach leverages the power of pre-trained models while allowing fine-tuning only on the top layers specific to the task at hand.

The other model is the Custom Convolutional Neural Network (CNN), which does not utilize any pre-trained models. This allows for better comparison and research by building a model from scratch. Before the training of the model, it is necessary to select and set the parameters in the training process, so as to make the model training effect better and get a better result. The following is the selection of parameter Settings for two models.

1)Optimiser:Both models use Stochastic Gradient Descent (SGD), which is one of the most common optimisers for training deep learning models, especially with a reasonably large dataset. Its benefits include simplicity and low memory requirements. However, SGD can have slower convergence rates compared to more advanced optimisers, and typically requires careful tuning of the learning rate. I experimented with numerous other optimisers (Adam, Adadelta, Adagrad), but they all got stuck in local minima despite being tested with numerous learning rates during training, and demonstrated worse loss and accuracy metrics than SGD.

2)Epochs and Batch Size:Both models were also assigned 500 epochs and a batch size of 32. However, due to the inclusion of an early stopping program, the 500 epochs serves more as an excessive placeholder to allow for the lowest possible loss to be found before finishing training. A batch size of 32 helps with avoiding memory issues, and sped up convergence. This relatively small batch size could have introduced some of the visible noise in the training curves, though.

3)Loss Function: Sparse categorical cross-entropy was employed in both models, which is appropriate when dealing with integer labels, as is the case in the dataset. It computes the cross-entropy loss between true labels and predicted labels effectively, without me having to convert the labels into tensors as would be required by standard categorical cross-entropy.

Unfortunately, it can be sensitive to label noise or mislabeled data points, if there are any present in the dataset.

4)Activation Function: Both models employed the rectified linear unit (ReLU) as their activation function for hidden layers, but used softmax for the output function. ReLU is one of the most common activation functions, as it introduces non-linearity and helps the network learn complex patterns. It also avoids the vanishing gradient problem (when gradients used to update the network become extremely small) and is computationally efficient. Softmax was the obvious choice for the output activation function, as it can effectively handle multiple labels. For the hidden layers, a ReLU variant such as leaky ReLU or parametric ReLU may have been beneficial due to their immunity from the so-called 'dying ReLU problem'; this could be looked into in a future investigation.

5)Learning Rate: For the transfer learning model, the learning rate was set to 10-4, while the learning rate was 10-3 for my custom CNN model. A smaller learning rate can make the training more stable and allow the model to converge to a good solution, which is why a small magnitude was chosen for both. Despite this, if the learning rate is too small, training can be slow, and the model might get stuck in local minima. This process was largely trial and error.

After selecting the hyperparameters for the models, you can start training the two different models separately. Figure 8 displays the changes in accuracy and loss function of the transfer learning model during the training process as the epochs progress. It is evident that the training of this model stopped after around forty epochs. This indicates that the magnitude of change in the loss function significantly decreased with increasing epochs, implying that further training is unlikely to yield significant improvements in performance.

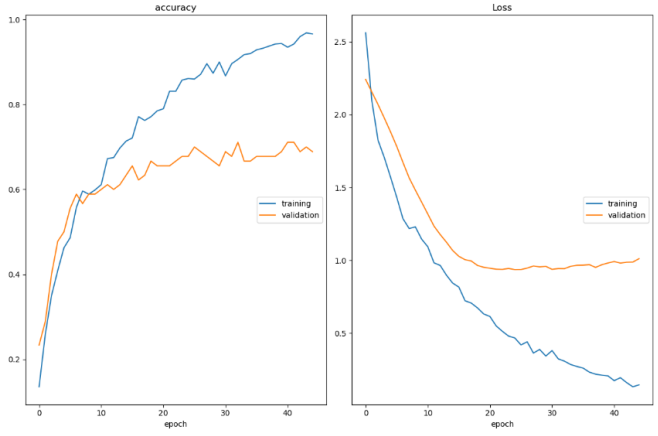


Fig. 8. Training process of transfer

Figure 9 illustrates the changes in accuracy and loss values of the custom model during the training process as the epochs progress. Compared to the transfer learning model, the custom model required almost double the number of training epochs. However, the custom model achieved better results in terms of accuracy and loss values at the final stopping point, especially from around epoch 50 onwards. At this point, the accuracy and loss values on the validation set started to plateau, indicating that further training would not significantly improve the performance.

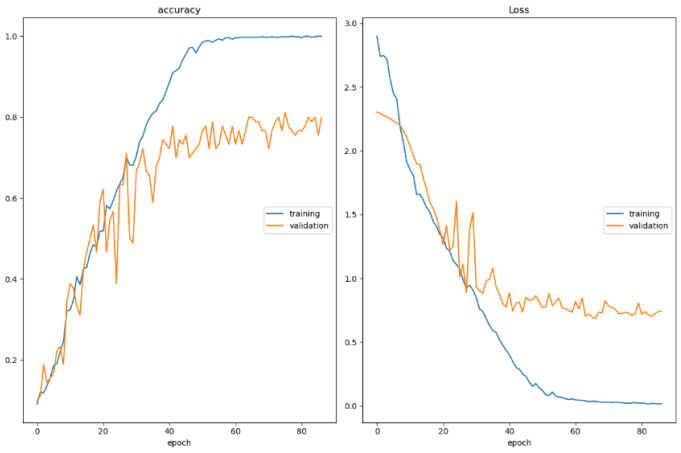
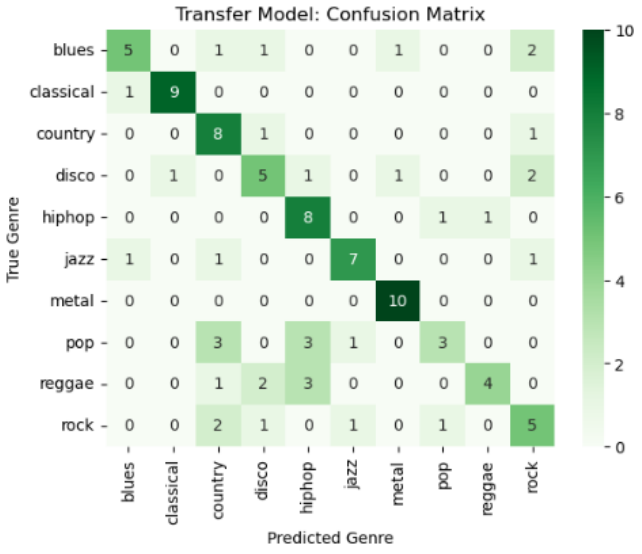


Fig. 9. Training process of CNN



Transfer Model
Training Accuracy: 97.875%
Train Loss: 0.1956

Validation Accuracy: 67.7778%
Test Loss: 0.9359

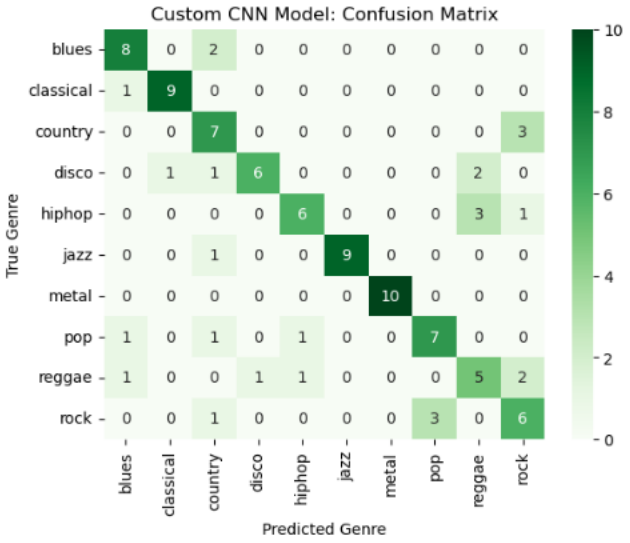
Testing Accuracy: 64.0%
Test Loss: 1.2608

Fig. 10. Confusion matrix of transfer

After training both models, you can assess their final performance using a confusion matrix. The confusion matrix

is a table used to evaluate the performance of a classification model. Each row represents the predicted results of the model, while each column represents the actual class labels. It provides a visual summary of the relationship between the model's predictions and the true labels.

Figure 10 presents the confusion matrix of the transfer learning model, along with its accuracy on three different datasets. It shows that the transfer learning model achieved a training accuracy of 97.9% and a testing accuracy of 64.0%. This suggests the possibility of overfitting, but the use of early stopping indicates that the model stopped training when the validation loss reached its lowest point.




```
Custom CNN Model
Training Accuracy: 99.875%
Train Loss: 0.0071

Validation Accuracy: 78.8889%
Test Loss: 0.6852

Testing Accuracy: 73.0%
Test Loss: 0.9799
```

Fig. 11. Confusion matrix of CNN

Meanwhile, the confusion matrix reveals that the model performed best in predicting the "Metal" genre (100% accuracy) and worst in predicting the "Pop" genre (30% accuracy). This could be because metal music has distinct characteristics in terms of sustained volume and frequency in the 30-second clips, making it easier to identify compared to other genres. On the other hand, "Pop" music is a broad category, making it difficult for the model to recognize patterns among the provided songs.

Figure 11 displays the confusion matrix of the custom CNN model and its accuracy on three different datasets. It shows that the custom CNN model achieved a training accuracy of 99.9% and a testing accuracy of 73.0%. Similarly, this suggests the possibility of overfitting, but the early stopping mechanism helps mitigate this issue.

The confusion matrix of the custom CNN model appears to be relatively linear, with only a few areas of mispredictions, primarily concentrated within the rock, hip-hop, and pop genres. These genres, especially pop music, often have ambiguous definitions, which can explain the decreased accuracy. Similar to the transfer learning model, it maintains a 100% accuracy in predicting metal music but experiences a drop to 50% accuracy in reggae music.

Finally, by comparing the confusion matrix and the accuracy of the three different classes, the CNN model using the custom base seems to be the "better" model, and here is how the CNN model can be used to classify a "random" song generated in the dataset, or at least shows how it can be applied to a new 30 second song clip.

```
1/1 [==============================] - 0s 60ms/step
Predicted genre: metal
Actual genre: metal
Filepath: metal/metal.00020.wav
```

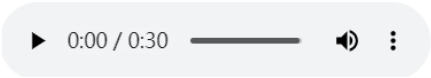


Fig. 12. Predicting Music Genre

V. Conclusion

Overall, the CNN model with the custom base demonstrates superior performance in terms of metrics, exhibiting better generalization and achieving 9% higher accuracy along with a lower loss of 0.2809 when handling unseen data. The results clearly indicate that the custom CNN model outperforms the transfer learning model utilizing a pre-trained InceptionV3 base. This can be attributed to the absence or scarcity of spectral images in the ImageNet dataset, resulting in more accurate predictions when using custom bases. Additionally, freezing the weights underlying the transfer model restricts its potential adaptation to new data, thereby preserving computational efficiency and retaining the pattern recognition capabilities of preset ImageNet weights.

Given the highly variable and subjective nature of music genres, I believe that small-scale neural networks like mine have limitations in achieving high accuracy in genre prediction. I consider achieving an ideal result with an accuracy of over 80% in testing to be challenging.

Overall, I believe this research has achieved quite successful results in accomplishing its goals. Despite the occasional ambiguity in music genre identification, I am satisfied with the state of the two models I have generated.

References

[1] Tzanetakis, G., & Cook, P. (2002). Musical genre classification of audio signals. *IEEE Transactions on speech and audio processing*, 10(5), 293-302.

[2] Yang, L. C., Chou, S. Y., & Yang, Y. H. (2017). MidiNet: A convolutional generative adversarial network for symbolic-domain music generation. *arXiv preprint arXiv:1703.10847*.

[3] Pons, J., Nieto, O., Prockup, M., & et al. (2017). End-to-end learning for music audio tagging at scale. *arXiv preprint arXiv:1711.02520*.

[4] Mossel, E., Neeman, J., & Sly, A. (2018). A proof of the block model threshold conjecture. *Combinatorica*, 38(3), 665-708.

[5] Dieleman, S., & Schrauwen, B. (2014). End-to-end learning for music audio. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 6964-6968).