Music Genre Classification using Transfer Learning on Spectograms

Alin Navas*
School of Medicine
UCD
Ireland
alin.navas@ucdconnect.ie

Abstract

This paper attempts to classify different 30-second audio clips into 10 music genre classes. We first use several preprocessing steps to convert the audio files into images and then attempt to perform the classification task initially using a baseline CNN model and then using pre-trained models to gauge any performance improvement.

1 Introduction

1.1 Dataset

The GTZAN dataset stands as the most commonly utilized public dataset for assessing music genre recognition (MGR) in machine listening research. Compiled between 2000 and 2001, the dataset comprises files sourced from a diverse range of origins such as personal CDs, radio broadcasts, and microphone recordings, aimed at encompassing various recording conditions. The dataset was available on Kaggle. There were audio files from 10 different classes containing 100 files in each class totalling 1000 files.

1.2 Rational for study

The purpose of this study is to explore methods for accurately classifying audio clips into different music genres. By utilizing various preprocessing techniques and machine learning models, including both baseline and pre-trained models, the study seeks to understand the effectiveness of these approaches in improving classification performance. This research contributes to the advancement of music genre recognition technology, which has applications in recommendation systems, music search engines, and other related fields.

2 Related Work

Multiple studies have been performed that use preprocessing methods like Mel Spectrogram, Mel Frequency Cepstral Coefficients, STFT (Short Time Fourier Transforms), Spectrogram and Chromograms to improve performance of audio classification tasks. Some of which are mentioned below. A study focusing on detecting environmental noise detection processed the audio data using Mel spectrogram model and MFCC resulting in 96.7 percent accuracy.[1] Another study aimed to develop an accurate and automatic respiratory sound classification system to aid in detecting and monitoring respiratory diseases. Three different feature extraction techniques—short-time Fourier

*MBBS, MSc

Preprint. Under review.

transformation (STFT), Mel spectrograms, and Wav2vec—are compared, along with three classifiers, including pre-trained ResNet18, LightCNN, and Audio Spectrogram Transformer. The key focus is on benchmarking these feature extraction techniques, with the STFT and fine-tuned ResNet18 network achieving notable results.[2]

This article introduces a method for automatic speech/music classification using chromagram textural and spectral features. The chromagram textural feature set involves transforming audio into chromagram images and extracting uniform local binary pattern textural descriptors. Chroma spectral features utilize novel chroma bin features exploiting music tonality.Evaluation on SS, GTZAN, and MUSAN databases shows the advantage and suitability of both chroma spectral and visual features for classification.[3]

Another paper investigates the performance of pre-trained Convolutional Neural Networks (CNNs) for sound classification, focusing on transfer learning. Initially designed for image classification, CNNs have been extended to sound classification.[4]

So I have attempted to use these 5 feature extraction methods and then used transfer learning to obtain an adequately performing model.

3 Experimental setup:

3.1 Feature extraction

The data downloaded was in an audio format and was preproceesed using feature extraction methods like Mel Spectrogram, Mel Frequency Cepstral Coefficients, STFT (Short Time Fourier Transforms), Spectrogram and Chromograms.

The Short-Time Fourier Transform (STFT) computes discrete Fourier transforms (DFT) over short overlapping windows, representing a signal in the time-frequency domain.

A Spectrogram visualizes the intensity of a signal over time at various frequencies, providing information on how the signal's energy is distributed across the frequency spectrum over time.

A Mel Spectrogram is a logarithmic transformation of a signal's frequency, based on the Mel scale. This transformation ensures that equal distances on the Mel scale correspond to equal perceived distances to humans.

Mel Frequency Cepstral Coefficients (MFCCs) are derived from the Mel spectrogram by applying Discrete Cosine Transformation (DCT) to create a spectrum over Mel frequencies, capturing the signal's spectral characteristics.

A Chromagram is a sequence of chroma features, each expressing how the representation's pitch content within a time window is spread over the twelve chroma bands or pitches. It provides information about the distribution of musical pitch classes over time.

These images were stored inside individual folders inside a parent folder called data pre. However, the folder structure was incompatible with the image data generator package in TensorFlow as there were no train, test, and validation folders with subfolders containing class labels. So this had to be created using custom code that split the images from each class into train, validation, and test sets. This was done in a 70-15-15 split due to the relatively low sample size of 100 images per class totaling 1000 images.

3.2 Transfer Learning

Transfer learning using the 4 models (EfficientNetB0,InceptionV3, ResNet50, VGG16) was then utilized on all 5 preprocessed audio types and their performance was assessed. The top-performing combinations were selected for further hyperparameter tuning to identify optimal hyperparameters. The models used were:

EfficientNetB0: EfficientNetB0 is a convolutional neural network architecture that achieves state-of-the-art performance with high efficiency. It uses a compound scaling method to balance between depth, width, and resolution, resulting in networks that are both accurate and efficient in terms of computation and memory usage.[5]

InceptionV3: InceptionV3 is a deep convolutional neural network architecture that uses inception modules to capture features at different scales. It employs various kernel sizes within the same layer to enhance feature extraction, allowing for efficient use of computation resources.[6]

ResNet50: ResNet50 is a deep residual network that addresses the degradation problem of deep neural networks by introducing skip connections or shortcuts. These shortcuts allow the network to learn residual mappings, making it easier to optimize and train deeper networks.[7]

VGG16: VGG16 is a convolutional neural network architecture known for its simplicity and effectiveness. It consists of 16 layers with small 3x3 convolutional filters and max-pooling layers, followed by fully connected layers. VGG16 is widely used for image classification tasks due to its straightforward architecture and good performance.[8]

3.3 Hyperparameter tuning

The selected combination of Preprocessing step and pre-trained model was then subjected to 3 rounds of hyperparamter tuning.

In Round 0 of the experiment, the primary hyperparameter under investigation was selecting the number of trainable layers in the pretrained model. The fixed hyperparameters included the number of fully connected layers , dropout rate, and activation function. Nuisance hyperparameters, such as learning rate and the number of neurons in the first fully connected layer, were varied across different values. The study evaluated the effect of varying the number of trainable layers across a range of neuron counts and learning rates.

Round 1- The scientific parameter for this round was the number of layers and usage of batch norm. The nuisance hyperparameters were dropout value, number of neurons per layer, learning rates, and different activation functions. The fixed hyperparameter was the number of trainable layers for the pre-trained models.

Round 2 - The scientific parameter for this round was the number of neurons in each layer, dropouts in each layer, and activation function. The fixed hyperparameters were the number of trainable layers, usage of batch norm, and number of layers. The nuisance hyperparameter is the learning rate.

4 Results

4.1 Preprocessing method - Pretrained model combination selection

F1 score was selected as the evaluation metric even though classes were balanced. The results of the first round of experiments are displayed below:

Table 1: EfficientNetB0

Preprocessing	F1 Score
chroma_imgs	0.0216
mfcc_imgs	0.0216
mel_spectrogram_imgs	0.0187
spec_imgs	0.0187
stft_imgs	0.0187

Table 2: InceptionV3

Preprocessing	F1 Score
chroma_imgs	0.2935
mfcc_imgs	0.4161
mel_spectrogram_imgs	0.3999
spec_imgs	0.4562
stft_imgs	0.2969

Table 3: ResNet50

Preprocessing	F1 Score
chroma_imgs	0.0160
mfcc_imgs	0.0710
mel_spectrogram_imgs	0.0160
spec_imgs	0.1308
stft_imgs	0.0113

Table 4: VGG16

Preprocessing	F1 Score
chroma_imgs	0.2831
mfcc_imgs	0.2980
mel_spectrogram_imgs	0.4042
spec_imgs	0.3780
stft_imgs	0.0215

Among these ResNet50 and EfficientNetB0 performed poorly for all the preprocessed audio types so these 2 models were not considered for further hyperparameter tuning.

For InceptionV3, the top-performing feature extraction technique was spectrograms and for VGG16 the best-performing technique was mel spectrograms so these combinations were selected for further evaluation.

4.2 Hyperparameter tuning results

We will look at the hyperparameter tuning results for the VGG16- mel spec combination first : After round 0 of experimentation with the objective value set as validation accuracy, we obtained a score of 0.6171875 with hyperparameters as 'num_layers': 1, 'num_neurons': 1765, 'learning_rate': 0.00129. Keeping this in mind, the number of trainable layers was fixed at -1 layer or the last layer only and the round 1 of experiments were performed. This time the val accuracy fell a bit to 0.609375 with the hyperparameters set as 'num_layers': 1, 'layer_size': 409, 'dropout_rate': 0.3736, 'use_batch_norm': True, 'activation': 'selu', 'learning_rate': 0.00010.

The fall in val_accuracy and inconsistency among the hyperparameters led me to think that the bayesian optimisation technique was not the best approach and instead a quasi random approach like the Quasi Monte Carlo(QMC) should be used first to more uniformly sample the entire search space and then based on the findings, another bayesian approach with a more narrow search space could be deployed. But unfortunately, the results obtained from the QMC were quite different from the initial Bayesian study. The best val_acc was the same as the initial Bayesian study with a value of 0.609375 but the number of layers,which was a scientific hyperparameter in this round, had increased to 3.

To clear this up, round 2 of the experiment was run with 3 FC layers to see if this resulted in better performance than round 1 Bayesian results. This study resulted in a further drop in val_acc to 0.59375 with hyperparameters: 'layer_size_1': 94, 'layer_size_2': 3975, 'layer_size_3': 3413, 'dropout_rate_1': 0.184935, 'dropout_rate_2': 0.1727, 'dropout_rate_3': 0.1733, 'activation': 'swish', 'learning_rate': 0.00014.

So, the best approach would be to test the results obtained from both round 0 or 1.

The same set of experiments was repeated with the InceptionV3-spectogram combination.

Round 0 of the study resulted in val_acc of 0.5703125 with the best Hyperparameters being : 'num_layers': 31, 'num_neurons': 6845, 'learning_rate': 0.000576. However, the num_layers value obtained was very close to the search space boundary which was 35, so the study was repeated to increase the search space. So after looking at the model architecture the last 31 layers was identified to be the last mixed block the InceptionV3 layer so the 2 other mixed blocks above it was added into the search space. But after repeating round 0 of the trial (version 2) a similar result containing -31 layers was obtained. So this was fixed in round 1 of the experimentation.

The result of round 1 of experimentation is worse than round 0 with the val_acc falling to 0.5078125 with hyperparameters of value: 'num_layers': 3, 'layer_size': 8800, 'dropout_rate': 0.1760, 'use_batch_norm': False, 'activation': 'relu', 'learning_rate': 0.000334. The number of layers was fixed to 3 and batch norm was not included in the next round of studies.

The round 2 of studies is still ongoing

The optuna studies are behaving a bit erratically. After looking into the issue, I believe that the low sample size is the root cause of the this behaviour. On visualising some of trials using optuna's visualisation package, we can observe that there are multiple trials with the exact same peak performance so the optimization is facing issues finding a specific trial for furtherfine tuning the hyperparameters.

A baseline convolution neural network based model was also trained using optuna and a performance of 0.45 was obtained as validation accuracy with hyperparameters: 'num_conv_layers': 4, 'kernel_size': 4, 'num_kernels': 88, 'stride': 2, 'activation': 'swish', 'dropout_rate': 0.131, 'use_batch_norm': False, 'num_dense_layers': 2, 'layer_size': 309, 'dense_dropout_rate': 0.293, 'dense_use_batch_norm': True, 'num_neurons_per_layer': 94. Due to the low performance of 0.45, further studies were not performed.

4.3 Model Performance and Training and Validation curves

From table 5,we see that VGG in round 2 of experiments seems to perform the best. The training and validation curves of this model along with the trial are also displayed in figure 1 and 2.

Table 5: F1 scores for different models.

Models	F1 Score
VGG_r0	0.6756
VGG_r1	0.5887
VGG_r2	0.7116
InceptionV3_r0	0.4891
InceptionV3_r1	0.4526
InceptionV3_r2	0.3226

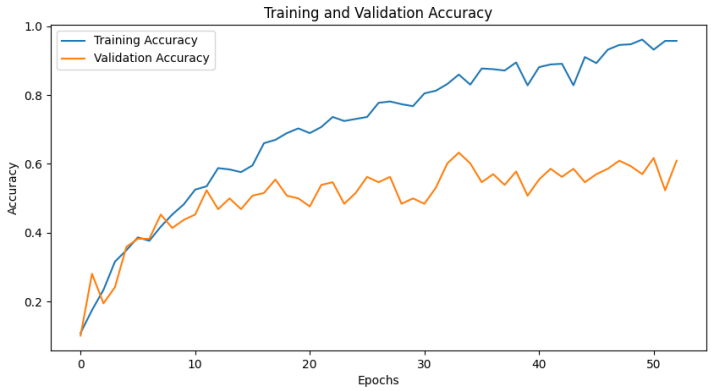

Figure 1: Training and Validation accuracy curves for VGG round 2

Figure 2: Training and Validation loss curves for VGG round 2

5 Conclusion and Future work

In my set of optuna studies, it seems like the VGG16 model performs best.

But I feel like one of the shortcomings of my approach was not focussing more on pre-processing especially splitting the 30-second audio files into smaller segments of 3,5,or 10 sec segments. I could have made this into a hyperparameter and checked the performance across a bunch of values. This

would have significantly increased the sample size thus improving the training as well as help provide more granularity for the optuna trials as there are now more validation samples. I first thought that I would use the width_shift_range option in the image data generator to create more augmented data that sort of resembled subsegments of audio data but this resulted in poorer performance so I didn't go ahead with it. But while going through some additional papers a week ago, I found a paper with significantly better performance on a similar dataset all because of that extra preprocessing step. I thought about implementing it but due to the SONIC queue, it takes 1-2 days on average for my code to start running and then another day to complete it (I am registered under School of Medicine, I think I am ranked lower on the queue). So I wouldn't have been able to do all the hyperparameter tuning again before the deadline. I have also decreased the trial size to 500 from 1000 for the later studies due to time constraints. The reason I went with 1000 in the first place, was because of the number of hyperparameters I was tuning at the same time and the wide search space.

Another shortcoming was not using pre-trained models that were trained on audio data like VGGish. I wasn't even aware that this was a thing until very recently. Again, due to computational and time constraints, I have decided not to go ahead with this model.

References

[1] Chaturvedi A, Yadav SA, Salman HM, Goyal HR, Gebregziabher H, Rao AK. Classification of Sound using Convolutional Neural Networks. In: 2022 5th International Conference on Contemporary Computing and Informatics (IC3I) [Internet]. 2022 [cited 2024 May 1]. p. 1015–9. Available from: https://ieeexplore.ieee.org/document/10072823

[2] Chen Z, Wang H, Yeh CH, Liu X. Classify Respiratory Abnormality in Lung Sounds Using STFT and a Fine-Tuned ResNet18 Network [Internet]. arXiv; 2022 [cited 2024 May 1]. Available from: http://arxiv.org/abs/2208.13943

[3] Birajdar G, Patil M. Speech/music classification using visual and spectral chromagram features. Journal of Ambient Intelligence and Humanized Computing. 2020 Jan 1;11.

[4] Tsalera E, Papadakis A, Samarakou M. Comparison of Pre-Trained CNNs for Audio Classification Using Transfer Learning. Journal of Sensor and Actuator Networks. 2021 Dec;10(4):72.

[5] Tan M, Le QV. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks [Internet]. arXiv; 2020 [cited 2024 May 2]. Available from: http://arxiv.org/abs/1905.11946

[6] Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z. Rethinking the Inception Architecture for Computer Vision [Internet]. arXiv; 2015 [cited 2024 May 2]. Available from: http://arxiv.org/abs/1512.00567

[7] He K, Zhang X, Ren S, Sun J. Deep Residual Learning for Image Recognition [Internet]. arXiv; 2015 [cited 2024 May 2]. Available from: http://arxiv.org/abs/1512.03385

[8] Simonyan K, Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition [Internet]. arXiv; 2015 [cited 2024 May 2]. Available from: http://arxiv.org/abs/1409.1556