Music Genre Classification by CNN

Mohamed Nasr

201801675

Abdelaziz Rashed

201700270

Zewail City

CIE 555 - Neural Networks and Deep Learning

Supervised by Dr. Mohamed Elshenawy

SPRING - 2022

1. Introduction

In this project, we implement how convolutional neural networks (CNN) can be used to classify music genres. We concentrate on the scenario where we have a limited computational and data budget and can not afford to use a large dataset to train. We begin with a well-known architectural structure in the scientific field and then employ transfer learning techniques. Various ways for fine-tuning, initializations, and optimizers will be implemented in order to determine how to choose the model that best suits the music genre classification.

2. Problem Definition

Music genres are a collection of descriptive terms that provide a music track (pop, jazz, metal, etc.). The purpose of genre classification is to predict the genre of music based on the audio input. We can automate the work of identifying musical tags that allows for making relevant material for both the user and the media company, such as music discovery and playlist creation.

To build this system, we should first extract acoustic features that are efficient estimators of the genres we are interested in, then by a single or multi-label classification or, in some cases, regression step. Typically, feature extraction employs a signal processing front-end to calculate significant features from time or frequency domain audio representations.

Next, the features are sent into the machine learning phase as input. Therefore, it is hard to determine which features are most important for performing each job. Modern Deep Neural Network (DNN)-based techniques combine feature extraction with decision making. As a result, the system may learn the essential features for each job while also learning to categorize them.

3. Method

3.1 Architecture of neural networks

In this project, we will begin with the CNN architecture described by Choi et al. in [1].

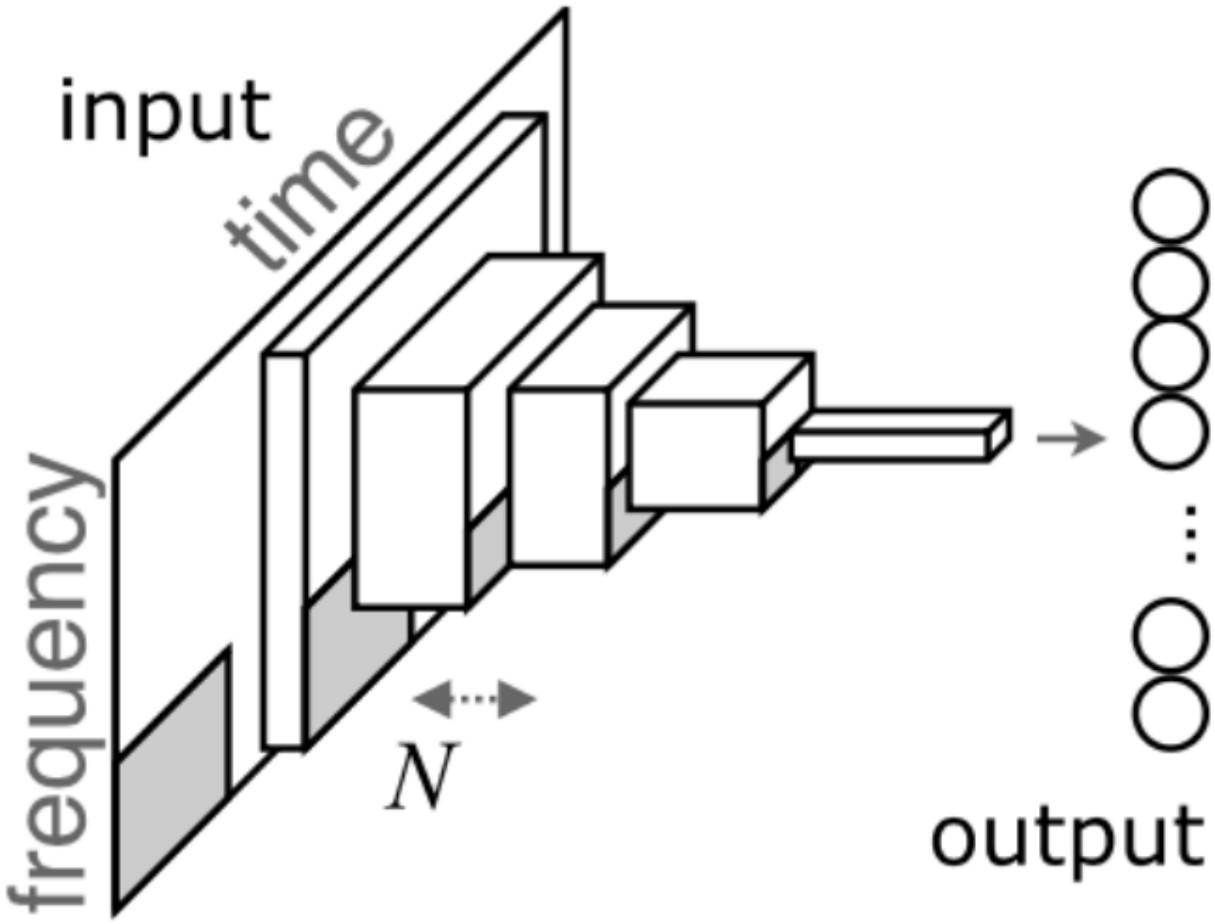

Figure 1. CNN Architecture

As shown in Figure 1, the first is a fully convolutional neural network with 5 convolutional layers of 33 kernels and max-pooling layers. At the last layer, the network lowers the size of feature maps to 1 * 1 such that each feature covers the whole input. By using incremental 2D subsamplings, this model provides time and frequency invariances at different scales. Being completely convolutional decreases the number of parameters significantly.

3.2 GTZAN Dataset

Tzanetakis et al[[2]] developed the GTZAN dataset. It is made up of 1000 audio records, each of them have a duration of 30 second audio samples, with 100 samples from each of the following music genres: Blues, Classical, Country, Disco, Hip Hop, Jazz, Metal, Popular, Reggae, and Rock. To be able to reflect a diversity of recording settings, the files were acquired in 2000-2001 from a number of sources, including individual CDs, radios, and audio samples.

Dataset Content from Kaggle:

- genres_original folder: The Tags of 10 genres types, for each one of them there are 100 audio, all of them are having a length of 30 seconds (the famous GTZAN dataset, the MNIST of sounds)
- Images_original folder: Which is the most important for our needs. We will use it directly. It is a visual representation for each audio file. One way to classify data is through neural networks. Because examples of neural networks (CNN - RCNN) 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.

3.3 Transfer Learning

Transfer learning [3] has shown to be particularly useful in the image processing field; it examines and proposes approaches for adapting a model learned in a large-scale database [4] to a smaller-scale database that performs effectively at jobs other than the one assigned as demonstrated in [5], [6].

Attempt to learn a high-performing model on a separate specific given data from a source given data (multi-class tags) (single class genres). This is referred to as domain adaptation under the transfer learning paradigm.

The two most popular practises, which we will implement, are:

1- Using the network to extract features. That is, the final fully-connected layer is removed, and the network is treated as a feature extractor. Once we've extracted all of the top features, we may add a classifier like SVM or Softmax to the dataset.

2- The network is being fine-tuned. This technique is predicated on not just replacing the network's classifier layer, but also retraining a portion or the entire network. Backpropagation allows us to change the weights of the pre-trained model in order to adjust it to the new data distribution. To avoid overfitting, it is often desirable to keep the initial layers of the network constant (or frozen) and simply fine-tune the deeper section. This is because the bottom levels of the networks capture general aspects that are comparable to many tasks, but the upper layers contain task and dataset relevant information, as proven in [7].

3.4 Pre-Trained models

We have initially four pre-trained models to be tested significantly. These four are:

1- EfficientNetB0 2- MobileNetV2

3- InceptionV3 4- ResNet152V2

To eliminate 2 of them, we found that ResNet152V2 took a lot of time before it completed its first epoch as it consumes a lot of computational resources . Then, we tried both

InceptionV3 and MobileNetV2 in a separate notebook. We found that InceptionV3 is better than MobileNetV2 in the validation accuracy..

So, we will implement only two pre-trained models for transfer learning:

1 - EfficientNetB0 2- InceptionV3

We will try each of them alone. Then we will choose the best one out of them to apply fine tuning.

3.5 Helpers

Our helpers are Tensorflow Keras callback functions at any model fitting. They are:

1- Model_save: to save the best weights during the run.

2- Early_stop: to specify the performance measure to monitor the trigger, then after triggered, it will stop the training process.

3- reduce_lr: to reduce the learning rate when the run stops improving.

In addition to these callbacks, there are some customization options in our code. For example, we used Adam optimizer. However, these customization details are out of scope of our project.

3.6 Work line

Firstly, we tried the default baseline CNN, then CNN with Drop-out. Secondly, Transfer learning for both of EfficientNetB0 and InceptionV3 pre-trained models. Finally- try to fine tune the best pre-trained model out of them.

4. Results

4.1 Baseline CNN

In figure 2 you can see the loss and accuracy for both training and validation sets of Baseline CNN. We noticed that the validation loss in the first 3 epochs is improving. Val accuracy is about 50 %. With the number of epochs increased, val accuracy goes for 62 % . However, val_loss did not improve from 1.30901. A call-back function that reduces learning rate is called. Best Val accuracy = 64 % while training accuracy near 100% (over fit). Dropout layers are a solution for overfitting.

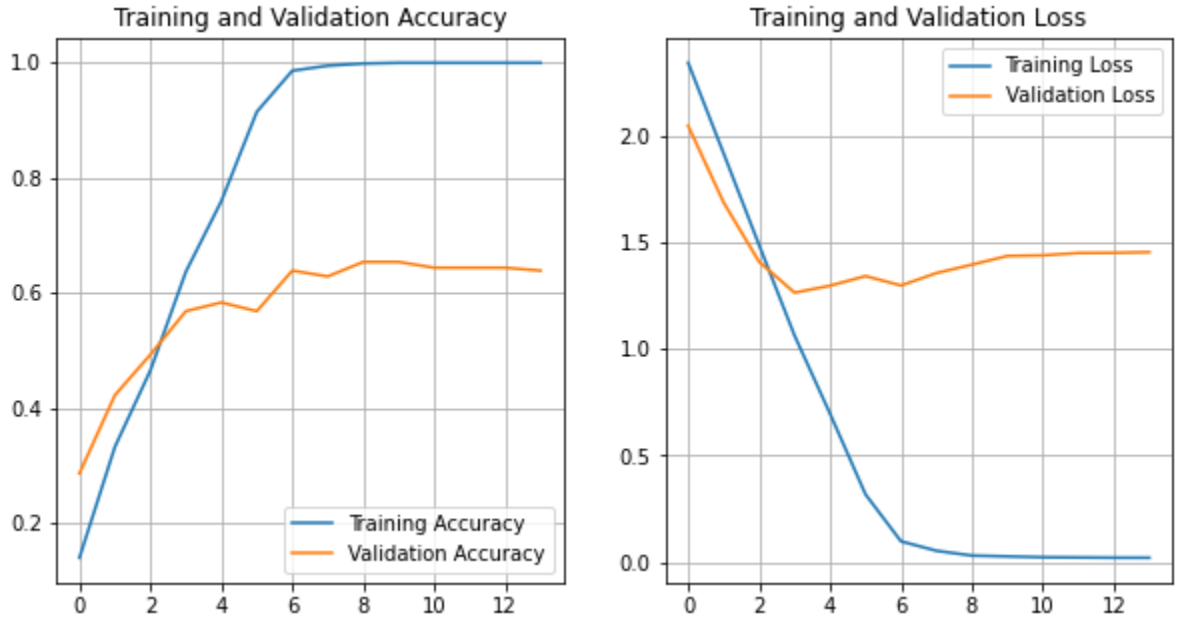

Figure 2. Baseline CNN

4.2 CNN with DropOut

In figure 3 you can see the loss and accuracy for both training and validation sets of CNN after adding three drop-out layers. Also after drop-out, the same problem of overfitting still exists. The accuracy increased a little bit in the 60 % range after adding three drop-out layers. We need a new solution for this since the problem appeared because of the small data set amount. So, transfer learning will be used to use a pre-trained model.

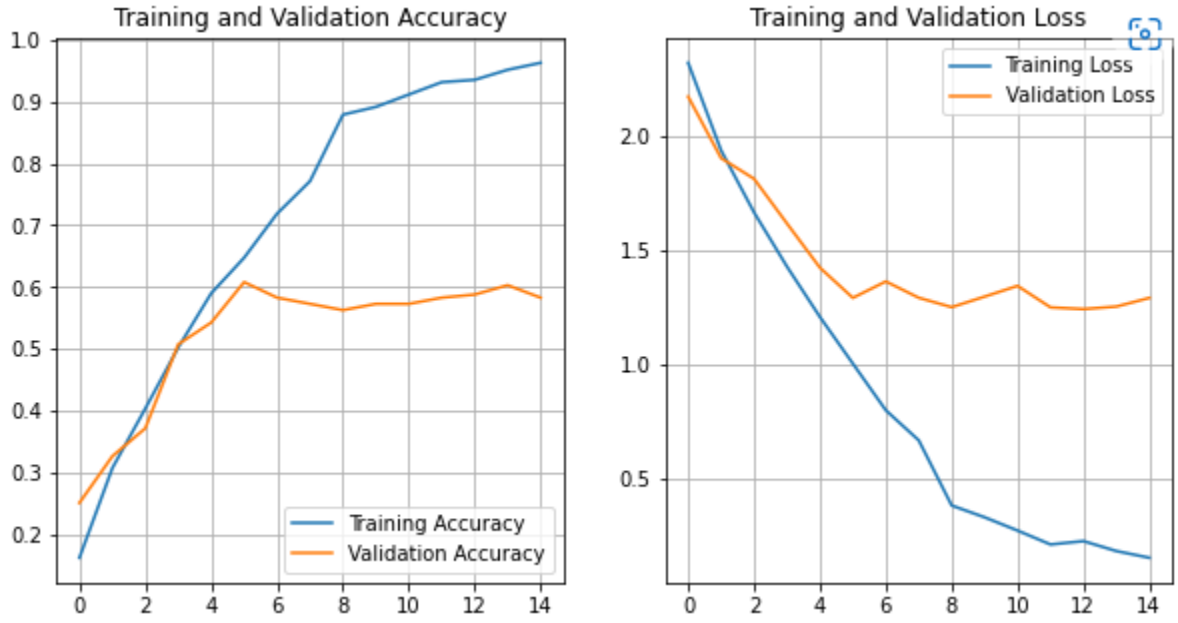

Figure 3. CNN with drop-out layers

4.3 Transfer Learning with InceptionV3

In figure 4 you can see the loss and accuracy for both training and validation sets of the pre-trained model InceptionV3. Validation Accuracy is around 73 % which started to be satisfying after using the transfer learning.

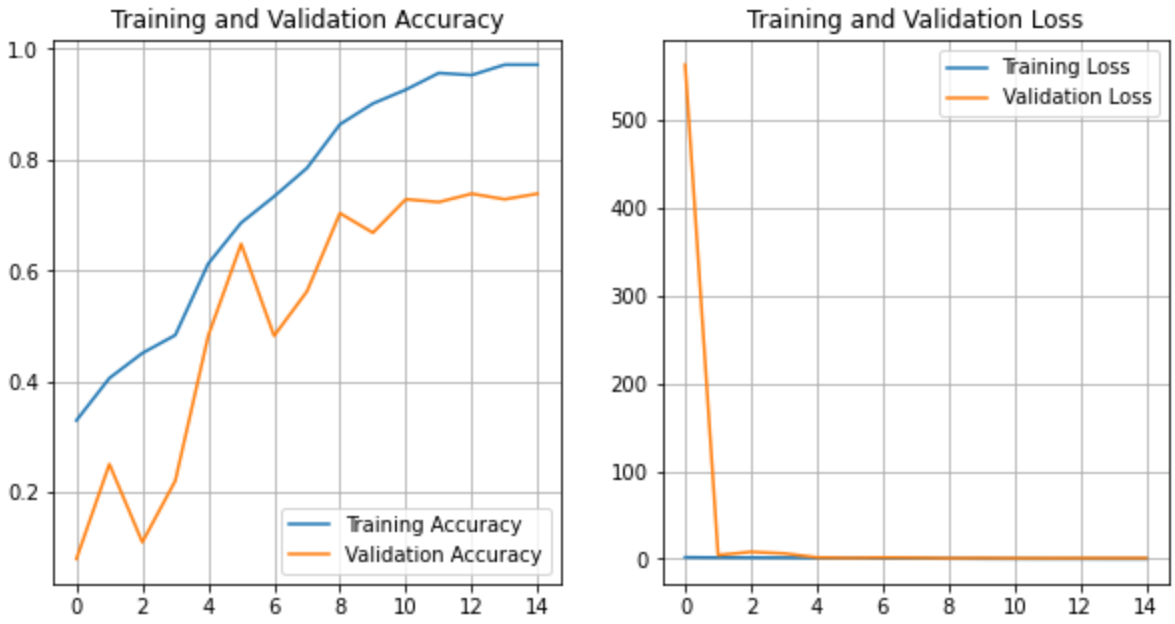

Figure 4. InceptionV3 pre-trained model

4.4 Transfer Learning with EfficientNetB0

In figure 5 you can see the loss and accuracy for both training and validation sets of the pre-trained model EfficientNetB0. It gives the best validation accuracy of 80 %.

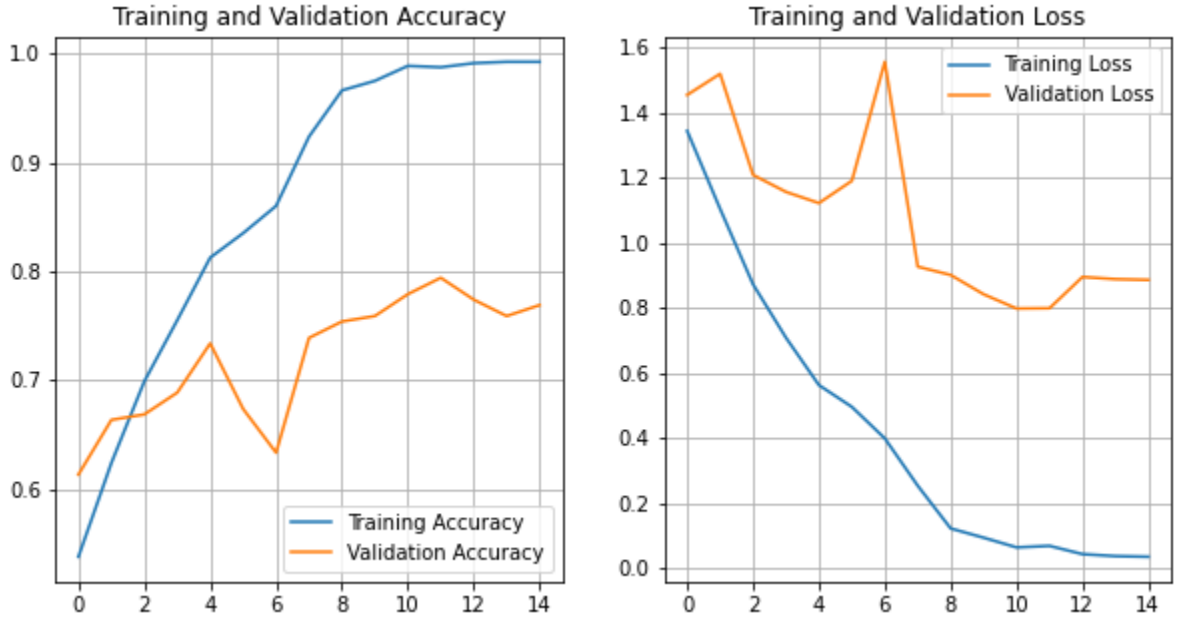

Figure 5. EfficientNetB0 pre-trained model

4.5 Transfer learning - EfficientNetB0 (the usual way of transfer learning)

The usual way of transfer learning involves freezing pre-trained layers. In figure 6 you can see the loss and accuracy for both training and validation sets of the usual way of transfer learning of EfficientNetB0 that involves freezing all pre-trained layers. It gives the best validation accuracy of 73%.

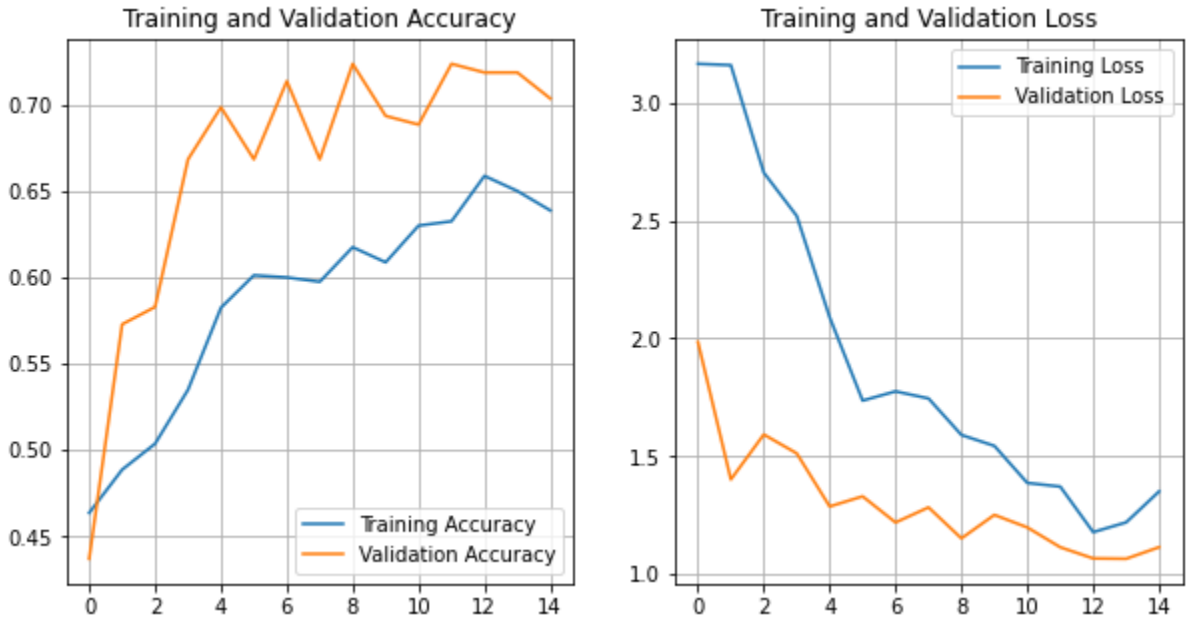

Figure 6. EfficientNetB0 freezing pre-trained layers

4.6 Fine-Tune - EfficientNetB0 (Unfreezing)

In figure 7 you can see the loss and accuracy for both training and validation sets of the usual way of transfer learning of EfficientNetB0 that unfreezes 100 of freezed pre-trained layers. It gives the best validation accuracy of 74%.

In figure 8 you can see the loss and accuracy for both training and validation sets of the usual way of transfer learning of EfficientNetB0 that unfreezes all of freezed pre-trained layers. It gives the best validation accuracy of 76%.

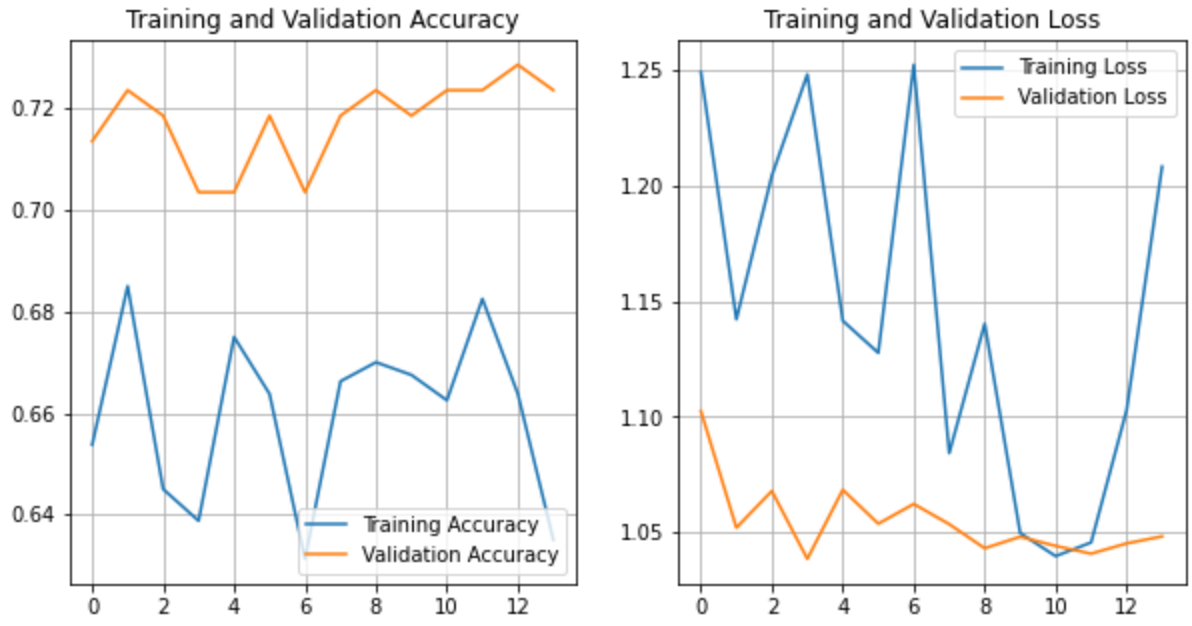

Figure 7. EfficientNetB0 Unfreeze 100 layers

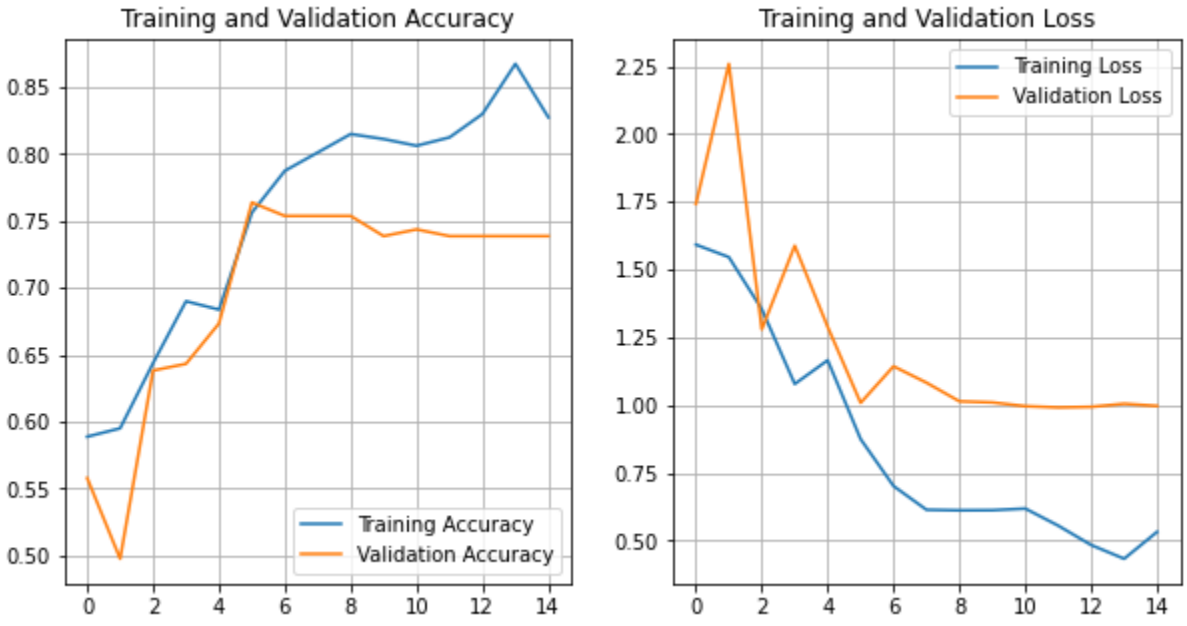

Figure 8. EfficientNetB0 Unfreeze all layers

5. Discussion

EfficientNetB0 with these customization is better in our case than InceptionV3 & MobilNetV2 models. We get the best validation accuracy that equals 80 % out of them.

Although there is an increase in validation accuracy,the problem of overfitting still appears as a result of lack of training data. Ordinary data augmentation may not be feasible for songs data like GTZAN, because we can not use typical transformations like rotation, zoom, and flipping because a spectrogram would be nonsense. Also, we can not use audio transformation because this will distort the original song.

6. Ethics

Although there are no clear harms that the application of music genre classification can cause, there are few things to consider:

Firstly, anyone who uses this program should avoid stealing some unique musical compositions out of genres without permission of the owner of this genre composition.

Secondly, this application should save copyrights for anyone who can use it in an illegal way.

References

[1] Keunwoo Choi, George Fazekas, and Mark Sandler, “Automatic tagging using deep convolutional neural networks,” arXiv preprint arXiv:1606.00298, 2016.

[2] George Tzanetakis and Perry Cook, Manipulation, analysis and retrieval systems for audio signals, Princeton University Princeton, NJ, USA, 2002.

[3] Sinno Jialin Pan and Qiang Yang, “A survey on transfer learning,” IEEE Transactions on knowledge and data engineering, vol. 22, no. 10, pp. 1345–1359, 2010.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, “Imagenet classification with deep convolutional neural networks,” in Advances in neural information processing systems, 2012, pp. 1097–1105.

[5] Matthew D Zeiler and Rob Fergus, “Visualizing and understanding convolutional networks,” in European Conference on Computer Vision. Springer, 2014, pp. 818– 833.

[6] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson, “Cnn features off-the-shelf: an astounding baseline for recognition,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2014, pp. 806–813.

[7] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson, “How transferable are features in deep neural networks?,” in Advances in neural information processing systems, 2014, pp. 3320–3328.