Music Genre Classification using Neural Networks

Mudigonda Himansh (AP19110010169)

Kuna Rajesh (AP19110010135)

Abstract

The music industry has grown multiple folds in the past couple of decades. With music coming up from young talented artists, with new genres making their way into the limelight, it is difficult to properly categorize the music into their respective genres. In this project, we try to implement cNN or convolutional neural networks and transfer learning to solve the problem. We convert the .wav file into an audio spectrogram and wavelet, then analyze it using the convolutional filter and make a deep learning model using TensorFlow and Keras. This deep learning model has about 8 million parameters to train for 10 different genres.

Introduction

Music is vocal or instrumentals (or both) combined in ways to produce harmony and expression of emotion. This rhythm is understood by the brain by a few specific parameters of the music like the tempo, musical instruments, timber, and so on. Music Genre Recognition is an important part of the research in Music Information Retrieval. A music genre is a conventional category that identifies some pieces of music.

Packages Used

os	Librosa
Pillow	Keras

Tensorflow	Shutil
Numpy	Random
Scipy	Pydub
Pydot	

Dataset

For the purposes of the project and to train the neural net, we make use of the GTZAN Dataset which is famous for MIR related projects. This dataset comprises 10 genres namely

Blues, Classical, Country, Disco, Hip-Hop, Jazz, Metal, Pop, Reggae, Rock.

Each Genre has 100 audio files (.wav) of 30 seconds. We can guess the genre just by listening to the first 4-5 seconds of the song. Hence, we can split each song into 10 audio files of 3 seconds each.

This means each genre has 1000 training examples. This implies that the entire dataset is not 10,000 samples.

Link to Dataset: https://github.com/ruhend/Music-Genre-Classification

Literature Review

Music recognition is a part of music information retrieval. There are many startups that concentrate on this field. The technology is always improving and the model is always training on new music and audio samples. This keeps the model not overfitted as the data is always new. For more accuracy in the recognition, we can change the tempo, and pitch, and add audio filters like distortion, reverb, and delay.

There is research going on in this field and it is a promising field. The literature is well discovered and also open to new and interesting ideas.

Base cNN Model

In this Model we start with a 2D Convolution layer with input size 32, 3 with a ReLU activation function. Then a MaxPool Layer. Then that is followed by a 32, 2 input 2D Convolutional Layer with a ReLU activation function. Followed by a MaxPool Layer again. Then we have a 64, 3 input 2D Convolutional layer, again followed by a MaxPool Layer. Then we apply a Drop-out layer with 40% of neurons getting deactivated. Then flatten is performed on the resultant output. Then it is followed by a Dense Layer with 128 neurons with ReLU activation. Finally, a 10 neuron output with softmax activation function, which is based on probability functions. We have selected a 10-neuron output layer as we have a 10-class classification.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 256, 256, 32)      896
 max_pooling2d(MaxPooling2D)  (None, 128, 128, 32)     0
 conv2d_1 (Conv2D)           (None, 128, 128, 32)      9248
 max_pooling2d_1(MaxPooling2D) (None, 64, 64, 32)      0
 conv2d_2 (Conv2D)           (None, 64, 64, 64)        18496
 max_pooling2d_2 (MaxPooling  (None, 32, 32, 64)       0
 2D)
 dropout (Dropout)           (None, 32, 32, 64)        0
 flatten (Flatten)           (None, 65536)             0
 dense (Dense)               (None, 128)               8388736
 dense_1 (Dense)             (None, 10)                1290
=================================================================
Total params: 8,418,666
Trainable params: 8,418,666
Non-trainable params: 0
```

Later, an Adam optimiser is applied, with loss function called Sparse Categorical Cross Entropy. We train the model with total of 500 epochs.

We save the weights as ‘500_epoch_simple_lr.cpkt’.

Transfer Learning based Model

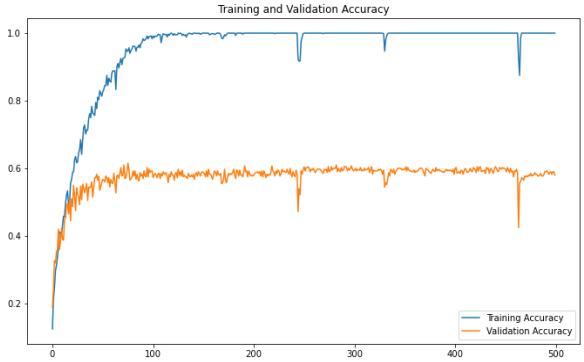
In this we use the MobileNetV2, with input shape 256,256,3. The Model is trained with Goblal Average Pooling 2D, with a 20% dropout applied on the model. The last layer with a dense layer with a 10 neuron output layer and a softmax activation function. It allows very memory-efficient inference and relies utilize standard operations present in all neural frameworks.

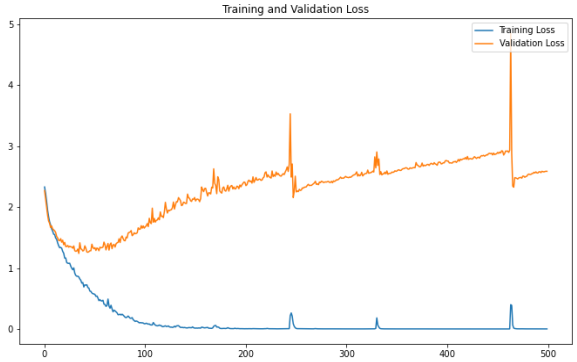
```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 mobilenetv2_1.00_224 (Funct  (None, 8, 8, 1280)       2257984
 ional)
 global_average_pooling2d_1   (None, 1280)             0
 (GlobalAveragePooling2D)
 dropout_2 (Dropout)          (None, 1280)             0
 dense_3 (Dense)              (None, 10)               12810
=================================================================
Total params: 2,270,794
Trainable params: 12,810
Non-trainable params: 2,257,984
```

Results

Basic cNN Algorithm

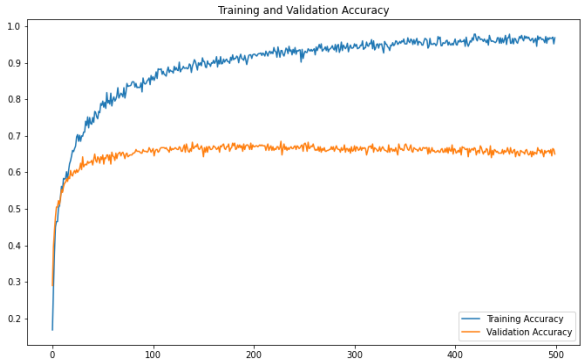
In the basic cNN Algorithm, we get a train accuracy of 100% with a 0.0000 loss in train dataset and a test accuracy of 58%, with 2.5869 loss in test dataset.

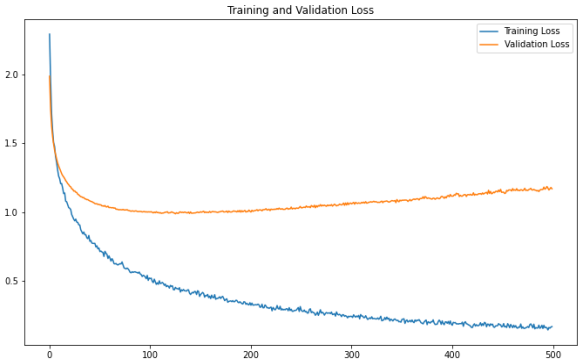




Transfer Learning Algorithm

With the transfer learning algorithm, we get a train accuracy of 96.83% with a 0.1663 loss in train dataset and a test accuracy of 65%, with 0.9683 loss in test dataset.





Conclusion

We have tested the Music Genre recognition using the Basic cNN Algorithm and Transfer Learning algorithms in this project. By gathering insights from the results, we can see the better training and testing results in transfer learning with a 7% hike in the accuracy.

Also, we can see the faster learning of the training dataset doesn't necessarily mean a higher accuracy in the test dataset.