Music Genre Detection Using Deep Neural Network on MEL Spectrograms

Achyut Karnani
33243387
ak19g21@soton.ac.uk

1. Introduction

Music has been a subject of research for a long time, and researchers have been trying to understand and differentiate between different genres of music. In this particular project, the aim is to classify music genres based on MEL spectrograms, which are formed by analyzing the audio clips of the music.This classification can be useful for organizing music libraries, recommending similar songs, and other applications.

To achieve this, the dataset used for the project was sourced from [1] website. This data for this was collected from various sources such as radios, CDs, microphone recordings, and so on, and contains a diverse range of music genres.

2. Understanding Dataset

The MEL spectrograms generated by the musics are converted to images for the classification. The following dataset contains data for 10 music genres. Named: blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, rock. These images are of size 3x432x 228. We reduce the size to 3x180x180 so that the sizes are symmetric across all genres, improve the computational and memory efficiency and prevent overfitting by getting rid of the noise.

3. Classification

We experimented with three different architectures of Neural Network to perform classification. Each of these methods/networks were run for 50 and 100 epochs to identify which is the most suitable model of them. We also experimented with two optimiser Adam and RMSOptimiser. CrossEntropyLoss function was identified as the suitable loss function for this problem.

The Adam optimizer was selected to optimize the loss function due to its adaptive learning rate, momentum, robustness, and regularization properties.Adam optimiser is a variant of stochastic gradient descent that uses adaptive learning rates for each parameter in the network, based on estimates of the first and second moments of the gradients. In RMSprop,a moving average of the squared gradient for each weight is used to split the gradient descent phase. As a result, the learning rate for each weight can be scaled based on the frequency of updates and the size of the slopes. The RMSprop method, in particular, keeps track of a rolling average of the squared gradient for each weight and makes use of this average to normalise the gradient before advancing in the direction of steepest decline.[2]

3.1. FeedFoward Neural Network

The first architecture comprised of only feedforward neural network. The input images, which are of size 3x180x180, are transformed into a flattened array with 97200 elements and then processed through two hidden layers before reaching the output layer. The first hidden layer has 128 neurons and the second hidden layer has 256 neurons. These hidden layers perform a series of transformations on the data, with each layer using a set of weights. During training these weights are updated to reduce the error between the actual value and predicted.

3.2. Convolution Neural Network

The second architecture comprised of Convolution Neural Network. The structure of the architecture used is given in the figure below 1. CNNs are chosen because they are specifically designed to learn and extract features and patterns from images using filters or kernels. These filters have weights that are learned during the training process. The output of one CNN layer is input to another, forming a chain of layers. Each layer contains a specific number of filters, as shown in the image below.

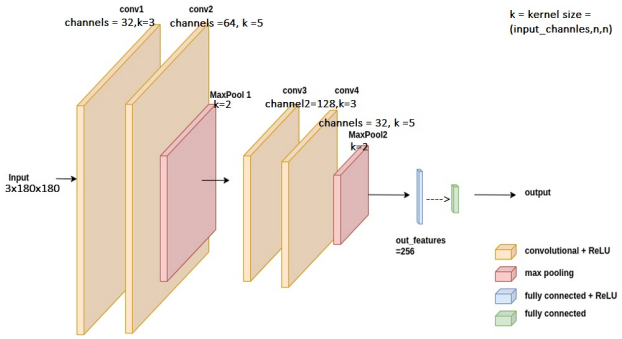


Figure 1: Neural Network Architecture

The stride for kernel is set to default (1, for normal conv, 2 for max pooling layers). Padding is also set to 0.

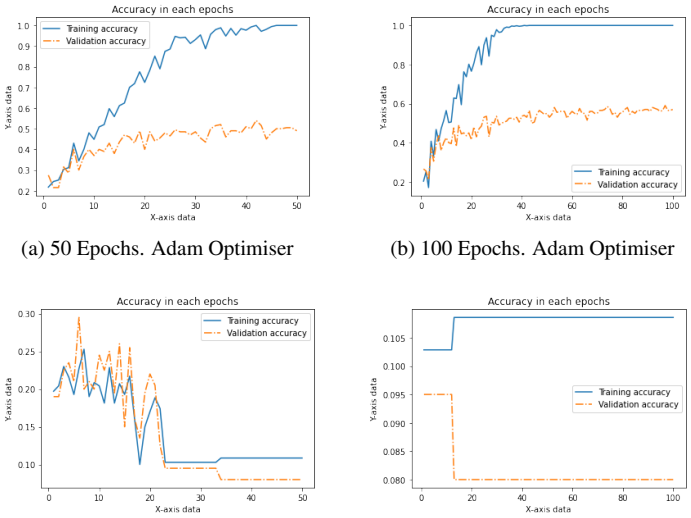
3.3. Convolution Neural Network with Batch Normalisation

For the next model we chose the same convolution architecture and setting as in 2nd Model but we add batch normalization layers after each convolution layer to improve the model performance.

4. Results and Observations

The Feedforward neural network began to overfit after the 6th and 5th when trained with Adam Optimiser . The last model after 50 and 100 epochs attained training accuracy of 100% and validation accuracy of 48% and 56%. When trained with RMS Optimiser, the model didn't overfit but under performed

significantly. It could achieve training accuracy of only 10% and validation accuracy of 7.9% after both 50 and 100 epochs.



(a) 50 Epochs. Adam Optimiser

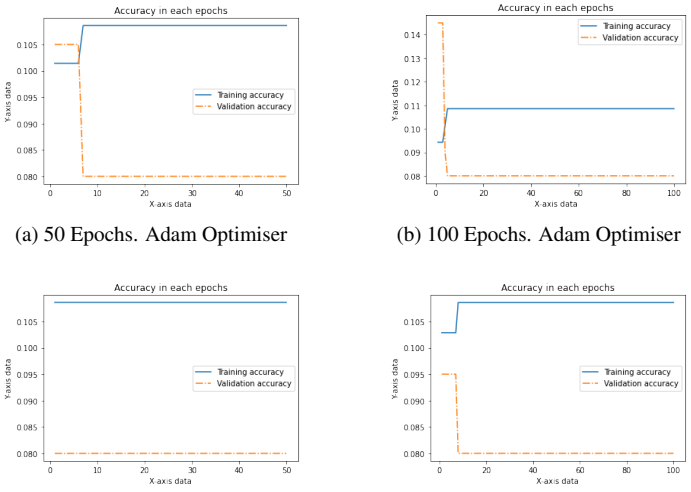
(b) 100 Epochs. Adam Optimiser

(c) 50 Epochs. RMSOptimiser

(d) 100 Epochs. RMSOptimiser

Figure 2: Feedforward Neural Network

The convolution neural network has a training accuracy 10% and a validation accuracy of 7% for the last model attained after 50 and 100 epochs wth Adam Optimiser. This indicates that the model is significantly underfitting. The training and validation accuracy are same when we used RMS Optimiser. This underfitting could be due to several factors, such as insufficient training data, a complex model, or the absence of regularization.



(a) 50 Epochs. Adam Optimiser

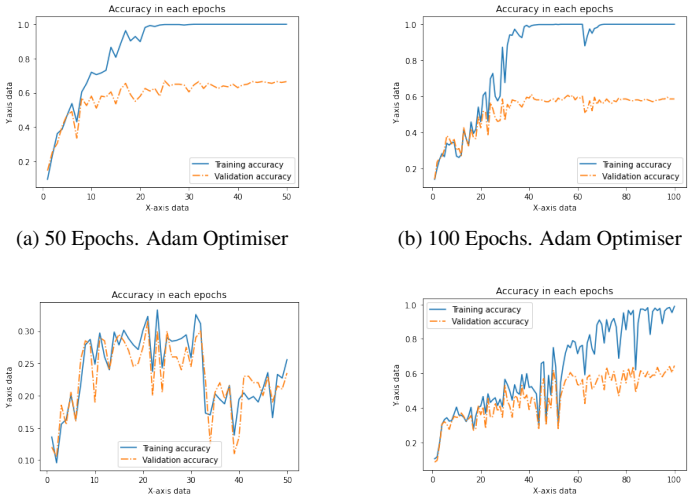
(b) 100 Epochs. Adam Optimiser

(c) 50 Epochs. RMSOptimiser

(d) 100 Epochs. RMSOptimiser

Figure 3: Convolution Neural Network - 1

After adding BatchNorm layers to the CNN architecture, we achieved the training accuracy of 60% and the validation accuracy to 50% by the 8th Epoch when trained with Adam Optimiser. After which the model began to overfit. To deal with this, we can use early stopping to get the best model. Similar pattern of training can be observed for the model when it was trained for 50 or 100 epochs. However, when trained with RMSOptimiser, the model did not overfit but the accuracy achieved was very low. 25% on training data and 23% on validation data.



(a) 50 Epochs. Adam Optimiser

(b) 100 Epochs. Adam Optimiser

(c) 50 Epochs. RMSOptimiser

(d) 100 Epochs. RMSOptimiser

Figure 4: Convolution Neural Network - 2 using BatchNorm layers

The final models obtained after 50 and 100 epochs were tested on testing data. These models chosen are the last models trained and may not be the best, but we can get those by early stopping. Most of the chosen models are overfitting while training, hence we do not expect different results on test data. Results on test data are given below:

	Adam		RMSOptimiser	
Model	Train_Acc	Test_Acc	Train_Acc	Test_Acc
FNN : 50 epochs	1.0	0.4646	0.1085	0.0808
FNN : 100 epochs	1.0	0.4343	0.1085	0.0808
CNN-1 : 50 epochs	0.1128	0.0808	0.1085	0.0808
CNN-1 : 100 epochs	0.1128	0.0808	0.1085	0.0808
CNN-2 : 50 epochs	1.0	0.6868	0.2557	0.1616
CNN-2 : 100 epochs	1.0	0.6666	0.9885	0.5353

Table 1: Results on training and testing set

5. Conclusion

The best results achieved so far was from the CNN architecture with BatchNorm layers with Adam Optimiser. The accuracy of this model can be increased using dropouts, adding more layers and by using more images. Furthermore, including image processing like normalisation, edge-detection, etc can further help in improving the accuracy of the model.

References

[1] GTZAN Dataset - Music Genre Classification https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification.

[2] Sanghvirajit. A Complete Guide to Adam and RMSprop Optimizer https://medium.com/analytics-vidhya/a-complete-guide-to-adam-and-rmsprop-optimizer-75f4502d83be, July 2021.