Music Genre Classification Using Convolutional Neural Network (CNN)

Bharat Dave [#1], Vinayak Chavan [#2], Mujahid Khan[#3], Dr. Varsha Shah[#4]

[1]Student, [2]Student, [3]Student, [4]Principal,

Computer Engineering Department, Rizvi College of Engineering.

Abstract: Music genres are categorical labels designed to differentiate the various pieces of music. Until recently, such classification was done manually, but now automatic music genre classifications can assist or replace human users in this process and contribute to a valuable framework in music information retrieval systems. Our designed system works on the above idea to extract features from audio files and learn from them to classify them across genres using a Convolutional Neural Network (CNN), all done on commodity hardware.

Keywords **– Music Genre, Classification, Machine Learning, Genre, Convolutional Neural Network**

I. INTRODUCTION

Musical genres are categorical labels created to characterize pieces of music. These characteristics typically are related to the instrumentation, rhythmic structure, and harmonic content of the music. Genre hierarchies are commonly used to structure the large collections of music available on the Web. Currently, most of the musical genre classification is performed manually. Automatic musical genre classification can assist or replace the human user in this process and would be a valuable addition to music information retrieval systems. Also, automatic musical genre classification provides a framework for developing and evaluating features for any type of content-based analysis of musical signals as music is the least worked data in the field of machine learning and AI. Automatic music genre classification is an application of artificial intelligence, more specifically machine learning that builds a system that predicts the genre of a song.

Automatic genre classification can be useful to solve some very interesting problems such as making song recommendations, finding similar songs, finding people who will like that particular song. Artificial Intelligence and automation are the core ideas that drove us to make this system. Our model is built on commodity hardware unlike the existing models built on high-end class hardware and gives us acceptable accuracy.

II. SURVEY EXISTING SYSTEM

Several models have been made to solve the problem of Music Genre Classification each one better than the last. The first of these papers is "Music Genre Classification Using Particle Swarm Optimization and Stacking Ensemble" [1], which tries to an audio file into of the 6 available 6 genres from the 'Thai Music Dataset' by extracting three features namely 'Rhythmic Content Features' which indicate the movement of signal over time, 'Timbral Texture Features' which use timbral features like spectral centroid, spectral flux, and so on and finally the 'Pitch Content Features' which use the pitch detection algorithms calculate pitch. The model classifies extracted features into various algorithms like KNN, Random Forest, Decision Trees, SVM, and Naïve Bayes. The best-acquired accuracy was 70%-79% which is quite low for 6 genres and using various such algorithms and takes too much time.

The second model is "Music Genre using Spectrogram" [2] which proposed an alternative idea to classifying music genre by converting an audio signal into a spectrogram and then extracting features from the said spectrogram. The model was trained on the 'Latin Music Dataset' which has 10 genres. The process converts the audio file into a spectrogram using the Short-Time Fourier Transformation (STFT) over 3 windows and then uses Gray Level Co-occurrence Matrix (GCLM) to extract the features finally training using Support Vector Machine (SVM) to get an accuracy of 67.2 %.

The third model is "Automatic Music Genre Classification based on Spectral Analysis of Spectral and Cepstral Features" [3] which divided the process into 3 parts- feature extraction, linear discriminant analysis, and information fusion. The feature extraction extracts three features namely Mel Frequency Cepstral Coefficients (MFCC), Octave Based Spectral Contrast (OSC), and Normalized Audio Spectral Envelope (NASE) which are then kept and discarded as per the use identified by the linear discriminant analysis and finally trained by creating a spectrogram and doing 10 fold cross-validation giving the accuracy of 90.6%. Their model performed even better than the ISMIR2004 Music Genre Classification Contest winner by a margin but at the cost of too much-specialized hardware and time for excellent accuracy.

III. PROBLEM STATEMENT AND OBJECTIVE

Problem Statement: Music plays a very important role in people's lives. Music brings like-minded people together and is the glue that holds communities together. Communities can be recognized by the type of songs that they compose, or even listen to. Different communities and groups listen to different kinds of music. This work exactly fulfills the above-mentioned requirement. This work will create a machine learning model that can predict the genre of the provided audio file.

Objective: 1. Develop a machine learning model using commodity hardware.
2. Train the CNN model to classify an audio file into one of the genres.
3. Allow individuals to get the genre of their specified audio file**.**

IV. PROPOSED SYSTEM

4.1 **Dataset**

The dataset used is the "GTZAN Genre Collection" created by George Tzanetakis which consists of audio samples of 10 different genres namely "Blues, Classical, Hip-hop, Jazz, Metal, Pop, Reggae and Rock", with each genre containing 100 (00000-00099) soundtrack files of 30 seconds each. Example of a filename: "Rock.00000.wav".

4.2 **Audio Features**

We used 4 audio features for the classification of the dataset. These are:

4.2.1 Mel Frequency Cepstral Coefficient (MFCC): MFCC are the coefficients of an MFC and the extraction procedure starts by windowing the signal, applying the Discrete Fourier Transform (DFT), taking the log of the magnitude, and then mapping the frequencies on a Mel scale. Then applying the inverse Discrete Cosine Transformation (DCT). The first few coefficients keep most information for a total of 39 coefficients per frame. We have used the first 13 coefficients in our model. The formula to convert the frequency to Mel scale has the following form:

$$m = 2915 \cdot log(1 + \frac{f}{500}) \qquad (4.1)$$

Also, the formula for the Fourier Transform is given as:

$$X(f) = \int_{-\infty}^{\infty} x(t) * e^{-i2\pi ft}\, dt \qquad (4.2)$$

4.2.2 Spectral Centroid: The brightness of a sound signal is determined by the spectral centroid. It can also be defined as the "Average Deviation of the rate map around its centroid" also known as variance around the centroid of the signal. The spectral centroid is higher where the sound is noisier and lower where the sound is warmer to hear. The formula to calculate the spectral centroid is given as:

$$SC_t = \frac{\sum_{n=1}^{N} m_t(n) * n}{\sum_{n=1}^{N} m_t(n)} \qquad (4.3)$$

4.2.3 Spectral Contrast: Spectral Contrast is defined as the difference between the peak and valley values of the spectrum. The formula for the peak value and valley value is given as:

$$\text{Peak Value} = \log\left(\frac{1}{\alpha N}\sum_{i=1}^{\alpha n} X_k, N + 1\right) \qquad (4.4)$$

$$\text{Valley Value} = \log\left(\frac{1}{\alpha N}\sum_{i=1}^{\alpha n} X_k, N - i + 1\right) \qquad (4.5)$$

Finally, the spectral centroid is calculated as the difference of the formula (4.4) and (4.5) as stated below:

$$\textit{Spectral Centroid} = \textit{Peak Value} - \textit{Valley Value} \qquad (4.6)$$

4.2.4 Chroma: Chroma-based features concern the musical notes that are played throughout a song. Information on the notes played most frequently throughout a song may be useful in classification for several reasons - for example, because different genres of music may tend towards different key signatures. Guitar-based music is more likely to be played in E or A than F or A#, because E and A chords are easier to play on standard-tuned guitars than are F and A# chords. Other styles of music may have no special leaning towards these particular keys or may lean toward other key signatures entirely. Therefore, all else being equal, if a song is detected to be in E or A, we might have an additional reason to believe that it is in a guitar-based style, unlike if it is detected to be in F or A#.

4.3 **Data Pre Processing**

Before we trained our model, we did pre-processing on the dataset so that it would reduce further complexity and save time during further training and testing processes. We divided the data into 3 sections, one each for training, testing, and validation. Primarily we separated all of these audio files with Testing having 80% audio files in Training, 19% in Validation, and 1% in testing folders respectively. Next, we converted files in Training, Testing, and Validation folders in Serialized data for easy storage. Two arrays were created after this, one for the data part upon which the model is trained, and the other is the part of the label where the output is checked. The data array consists of the file name and their features with their values in particular columns post-extraction an example of which is shown in table 4.1.

Table 4.1 Data Array

Sr. No.	Filename	Features							
		Mfcc 1	….	Mfcc 13	Chroma 1	….	Chroma 12	Spectral Centroid	Spectral Contrast
1	classical.00009.wav	-4.14	-1.4	-4.2	-3.40	-3.2	-4.47	+2.89	+5.19
2	jazz.00004.wav	+2.12	+3.2	+2.98	-5.98	-4.9	-5.33	-4.78	+1.19
3	metal.00031.wav	-1.17	-3.1	-5.18	+2.20	+4.33	+1.14	-4.41	-3.33

The second table is the labels table where we used one-hot encoding to encode the filenames with their outputs, essentially giving the row a value of 1 in their corresponding column and 0 in others so that the model checks for which column was the file had a value of 1 to find whether the predicted output was correct or not. The one-hot encoded labels tables for a few rows is shown below:

Table 4.2 Labels Array

4.4 **Model Architecture**

Sr. No.	Filenames	Classical	Rock	Pop	Hip-hop	Jazz	Reggae	Blues	Country	Disco	Metal
1	classical.00009.wav	1	0	0	0	0	0	0	0	0	0
2	jazz.00004.wav	0	0	0	0	1	0	0	0	0	0
3	metal.00031.wav	0	0	0	0	0	0	0	0	0	1

Model Architecture defines the logical connections of various layers used in the model creation and training. The various layers used in our model are:

4.5.1. Convolution layer: In this layer, we consider a certain number of filters and we take one filter and convolve them (slide them) around the entire image at the same time multiplying the pixel value of the image with the corresponding pixel value of the filter adding them up and dividing by the total number of pixels to get the output. So we will get the outputs equal to the number of filters we choose.

4.5.2. Rectified linear unit (Relu) layer: Relu transform function only activates a node when the output is above a certain quantity, if the input is below zero, the output will be zero only, when the input goes above a certain value, it has a linear relationship with the dependent variable. Hence in this layer, the values below zero will become zero and the values above zero will remain as it is in the output received from the Convolution layer.

4.5.3. Pooling layer: In this layer, we reduce the image stack to a smaller size. The pooling can be of the maximum, minimum, or average value. The steps for pooling are:

- Pick a window size (usually 2 or 3).
- Pick a stride/shift (usually 2).
- Walk your window across your filtered images.
- From each window, take the maximum/minimum/average value.

4.5.4. Fully connected (Dense) Layer: After passing the image through a stack of convolution, Relu and pooling the output arrives at the final layer known as the fully connected layer. Here final classification happens. Here we take our filtered and shrunk images and put them into a single list. In the list, there will be a certain value high for a particular genre which helps in its genre classification.

4.5.5. Dropout layer: The dropout layer randomly discards x% of the available values given as the input. The value of the x is specified by the user and is specified between the ranges 0.0 to 1.0.

4.5.6. Flatten Layer: The flatten layer converts the 2 dimensional 'm x n' array into a single dimensional 'm + n' array essentially flattening the data into a linear form.

4.5.7. Softmax Layer: This layer is used for more than 2 outputs and gives each output a value from 0 to 1, the addition of each of these outputs is also 1 and the output which has maximum value is the predicted output. The value of output given by softmax is determined by the number of nodes in its previous layer.

Our model architecture is as follows:

- Convolutional Layer [No. of Filters=48, Kernel size=5]
- Pooling Layer [Pool Size =2 , Type = Average]
- Relu Layer
- Convolutional Layer [No. of Filters=48, Kernel size=5]
- Pooling Layer [Pool Size =2, Type = Average]
- Relu Layer
- Convolutional Layer [No. of Filters=48, Kernel size=5]
- Pooling Layer [Pool Size = Type = Average]
- Relu Layer
- Flatten Layer
- Dropout Layer [Dropout Percentage= 50%]
- Dense Layer [Nodes =100]
- Dropout Layer [Dropout Percentage= 40%]
- Dense Layer [Nodes =10]
- Softmax Layer

V. HARDWARE AND SOFTWARE REQUIREMENTS

5.1 Minimum Requirements:

a) Hardware:

- CPU- Intel Core i3 5th gen/ AMD Ryzen 3 1300 or above.
- GPU- Nvidia GTX 1050 or above.
- RAM- 4 GB or above.
- HDD- 2GB free space.

b) Software:

- Windows 7/8/8.1/10 (32 bit/64 bit) or Ubuntu Linux or Mac OS.
- Python 3.1 with Librosa, Keras & Tensor flow libraries installed.

5.2 Recommended Requirements:

a) Hardware:

- CPU- Intel Core i5 8th gen/ AMD Ryzen 5 2600 or above.
- GPU- Nvidia GTX 1660 or above.
- RAM- 8 GB or above. HDD- 4GB free space.

b) Software:

- Windows 7/8/8.1/10 (32 bit/64 bit) or Ubuntu Linux or Mac OS.
- Python 3.1 with Librosa, Keras & Tensor flow libraries installed.

VI. RESULT AND DISCUSSION

Table 6.1: Metrics

Metrics Name	Minimum	Maximum	Average
Accuracy	50 %	65 %	57.5 %
Loss	16.0717	9.4065	12.7391
Precision	0.64	0.86	0.75
Recall	0.40	0.79	0.63

Table 6.1 depicts the metrics acquired post the fine-tuning of hyper parameters and getting the best possible values for commodity hardware. The displayed metrics are Accuracy, Error Rate, and Precision and Recall, each having minimum, maximum and average values.

Accuracy is the representation of how good our model predicted on the data provided and is given in range 0.000 to 1.00 and is multiplied by 100 to get required accuracy. An example would be 50% accuracy displayed as 0.500. The accuracy gives an average of 57.5 % by our model. Initially, the accuracy was severely less due to the model overfitting on the dataset since we did not have any dropout layers. Adding dropout layers with a ratio of 40% increased to a significantly better value which shifts between the acquired ranges. The model performed approximately the same at different numbers of convolutions with minute point differences in dropout and hence no new convolutions were added post that. We also decided on 400-700 as the best number of epochs to train our model for since below 400, the training was inadequate, and post 700 the accuracy was a constant value with no change.

Loss is the quantitative measure of deviation between the predicted value and the actual output. A lower loss means a model is learning far better. The loss like every other machine learning model begins at an extremely high value since our model is not yet trained and every result is predicted wrong. The loss decreased at a slow rate even after many epochs and hence we increased the number of convolutions to 48 for all three convolutions layers from the initial tried value of 12 and 24. This increase in convolutions made our model learn better and the dropout helped avoid overfitting since our training data just got bigger for a small dataset. Any increase in convolutions from 48 caused the model to over fit on training data and upon changing dropout at this point of time, the accuracy decreased drastically to below 40%. Hence we concluded that 48 was the best number of filters for the model.

Precision and Recall are the most important values for pattern recognition and classification model. Precision is the ratio between the correct predictions and the total predictions. In other words, precision indicates how good the model is at whatever it predicted. So if it predicted 10 things and it turned out to be wrong at three, the precision is 7/10 (0.7). The recall is the ratio of the right predictions and the total number of right items in the set. Recall indicates how good is the model at picking the right items by remembering the previous predictions. Since precision and recall are said to be co-related values, so if one increases then the other decreases. This was the case in our model too. The precision starts at the maximum value of 1 while recall was at 0. As our model learned better by adjusting the accuracy and loss as we changed the parameters like the number of convolutions and dropout, we got a good trade-off between precision and recall. The precision and recall are slightly dependent on the accuracy and total correct predictions and hence the same amount of optimization work for these.

Our future work would be to increase the accuracy by tuning the parameters and in turn, gets better metrics. One of the possible solutions is to train on only a few of the genres which are similar to each other in features and hence model would learn better than the current version.

REFERENCES

[1]Krittika Leatpantulak, Yuttana Kitjaidure, 6-8 March 2019, " *Music Genre Classification of audio signals using Particle Swarm Optimization and Stacking Ensemble"*, 2019 7th International Electrical Engineering Congress (iEECON), INSPEC Accession Number: 19228693.

[2] Yandre M.G Costa, Luiz S. Oliveira, Alessandor L. Koericb, Fabien Gouyoun, 16-18 June 2011, *"Music genre recognition using spectrograms"*, Institute of Electrical and Electronics Engineers (IEEE), Print ISBN:978-1-4577-0074-3, INSPEC Accession Number: 12177876.

[3]Chang-Hsing Lee, Jau-Ling Shih, Kyun-Min Yu, Hwai-San Lin, 28 April 2009, *"Automatic Music Genre Classification Based on Modulation of Spectral and Cepstral Features*", Institute of Electrical and Electronics Engineers (IEEE), Print ISSN: 1941-0077, INSPEC Accession Number: 10664199.

[4]Brian McFee, Python Librosa Documentation, *https://librosa.org*

[5]François Chollet**,** Python Keras Documentation, *https://keras.io*

[6]Steve Tjoa**, "***Notes on Music Information Retrieval", https://musicinformationretrieval.com*