A Project Report

on

Music Genre Classification using CNN and RNN-LSTM

Submitted in partial fulfillment of the

requirement for the award of the degree of

Bachelor of Technology in Computer Science and Engineering

GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Under The Supervision of

Mr. Lalit Sharma
Assistant Professor
Department of computer science and engineering

Submitted By

Naveen Kumar
18021011411/18SCSE1010166

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GALGOTIAUNIVERSITY, GREATER NOIDA,INDIA
DECEMBER, 2021

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the thesis/project/dissertation, entitled **"**MUSIC GENRE CLASSIFICATION USING CNN AND RNN-LSTM**"** in partial fulfillment of the requirements for the award of B.Tech submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of August, 2021 to December, 2021, under the supervision of Lalit Sharma Assistant Professor, Department of Computer Science and Engineering/Computer Application and Information and Science, of School of Computing Science and Engineering , Galgotias University, Greater Noida

The matter presented in the thesis/project/dissertation has not been submitted by me/us for the award of any other degree of this or any other places.

Naveen Kumar
18SCSE1010166

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Lalit Sharma
Assistant Professor

CERTIFICATE

The Final Thesis/Project/ Dissertation Viva-Voce examination of Naveen Kumar(18SCSE1010166) has been held on _____ and his/her work is recommended for the award of B.Tech in Computer Science And Engineering.

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator

Signature of Dean

(School of Computing Science and Engineering)

Date: December, 2021
Place: Greater Noida

Acknowledgement

With the ongoing revolution in Artificial Intelligence and Machine Learning where innovations are taking place at the blink of eye, it is possible to keep pace with the emerging trends. Excellence is an attitude that the whole of the human race is born with. It is the environment that makes sure that whether the result of this attitude is visible or otherwise
.
Preparing of the report provide a linkage between a student and the latest trends to develop an awareness.

I express my sincere thanks to Mentor Mrs Lalit Sharma Assistant Professor, Galgotias University, Greater Noida for her valuable guidance and feedbacks.

The author would like to express a deep sense of gratitude and thank our mentor, without whose permission, wise counsel and able guidance, it would have not been possible to carry out our research in this manner.

Finally, I express my indebtedness to all who have directly or indirectly contributing to make our research successful.

Name- Naveen Kumar

Abstract

“The beautiful thing is, music can be like a time machine. One song- the lyrics, the melody, the mood- can take you back to a moment in time like nothing else can.
Music is the glue that holds people together.” The objective is to find a better Machine Learning (ML) algorithm which can predict the genre of songs. Music Genre Classification has been a difficult yet encouraging assignment in the field of music MIR is one of the popular techniques which is used nowadays to get useable information from the music (audio signal). subtle qualities of audio musical data, recovering instructive and dependable features from audio signals is significant to the presentation of any music genre. In this, we use CNN and RNN to classify the music clips. CNN is a deep learning architecture and it has been widely used in pattern recognition literature.
CNN has strong amplitude to capture informative features from the variations of musical patterns with minimal prior knowledge provided. compare the performances of all these models and log their results in terms of prediction accuracies.

Keywords: Machine Learning Algorithm, Music information retrieval (MIR), Music Genre Classification, Covolutional Neural Network(CNN), Recurrent Neural Network

Contents

List of Table

List of Figures

Acronyms

B.Tech.	Bachelor of Technology
M.Tech.	Master of Technology
BCA	Bachelor of Computer Applications
MCA	Master of Computer Applications
B.Sc. (CS)	Bachelor of Science in Computer Science
M.Sc. (CS)	Master of Science in Computer Science
SCSE	School of Computing Science and Engineering

CHAPTER-1
Introduction

1.1 Introduction:

In today's digital world, there would hardly be any person who is not accessing Internet in their day-to-day lives. People are addicted to various websites and apps which have become an inseparable part of their lives. One such kind of apps are the music streaming apps such as Youtube Music, Spotify, itunes, wynk and many more. Since the music database of such apps is really huge and ever-increasing,
there is always a need to organize & categorize the music which is prevailing inside this database. Music Genre Classification is one of the most popular ways to achieve this. While listening to online music using these apps, sometimes the songs are not categorized correctly as per their music genre collection. This problem affects the mood of the user if they are listening to songs of a specific genre, which are being automatically categorized by the app. This problem can be resolved by using Machine Learning algorithms, which can improve the way of music genre classification. Some of the possible ML algorithms are traditional classification methods, HMM,Neural Network, etc.

1.2 Problem Formulation:

While listening to online music using music streaming apps, sometimes the songs are not categorized correctly as per their music genre collection. This problem impacts the mood of the user if they are listening to songs of a specific genre which are being automatically selected by the app. This problem can be resolved by using Machine Learning algorithms which can improve the way of music genre classification. Some of the possible ML algorithms are traditional classification methods, HMM, Neural Network, etc. In this research, we are planning to use CNN and RNN to enhance the accuracy of our genre classification prediction model.

1.2.1 Tool and Technology Used

Programing language	• Python
Tools	• TensorFlow • Jupyter Notebook • PyCharm • Visual studio code
Operating System	• Windows

Chapter 2
Literature Survey

Music Genre Classification is an area which has attracted the interest of many researchers. This section will provide details about some of the research work already done in this field.

Hareesh Bahuleyan, "Music Genre Classification using Machine Learning Techniques" Published in 2018.He proposed two different approaches to solve this problem. The first involved generating a spectrogram of the audio signal and treating it as an image.VGG-16 an CNN based image classifier, is trained on those images to predict the music genre solely based on spectrogram. The second approach consisted of extracting time domain and frequency domain features from the audio signals followed by training traditional machine learning classifiers XGBoost, based on
features.

Jeremy Reed and Chin-Hui Lee, "A Study on Music Genre Classification Based on Universal Acoustic Models" Published in 2006. The algorithm they had presented provides comparable results to past solutions of the genre classification problem. This paper was based on HMM modelling. According to their research "If genre classification is comparable to language recognition, then modelling HMMs in this way would equate to having a single HMM for an entire spoken document or entire language. Most speech applications use HMMs on the phonetic level and are therefore able to use syntactic rules to improve classification performance." This research
demonstrated that a similar approach may be possible for music, even though labelled training corpora are not in existence.

Thomas Lidy and Andreas Rauber, "Evaluation of Feature Extractors and Psycho-Acoustic Transformations for Music Genre Classification." Published in 2005. They performed a study on the contribution of psychoacoustic transformations in the calculation of Rhythm Patterns for efficient content-based music description. Numerous experiments have been arranged to identify the important parts in the feature extraction process.

Kaichun K. Chang, Jyh-Shing Roger Jang and Costas S. Iliopoulos, "On music genre classification via compressive sampling" Published in 2013.They proposed a CSbased classifier and verified its performance by a common dataset for music genre classification. Moreover, They had also explored the possibility of using multiple feature sets for improving the performance of genre classification. The experiments demonstrate that the proposed CS- based classification together with the use of multiple feature sets outperform quite a few state-of-the-art approaches for music genre classification. The success of the proposed CS- based classifier is attributed to CS's superb capability in feature extraction for generating parsimonious representation of the original signals.

As mentioned in “A Study on genre Classification supported Universal Acoustic Models” by Jeremy Reed and Chin-Hui Lee which was published in 2006, a collection of computer instructions may result in similar results because the ones received by the earlier solutions of the genre classification problem. These instructions were supported HMM Modelling. in keeping with their research if any classification is a dead ringer for language identification then modeling of HMMs would be able to classify one HMM for complete language.

HMMs are widely used by most of the speech applications on the phonetic level. Hence, it can be utilized to improve the performance of classification. It was also proved by this research that a similar approach can be taken for classifying music.

As proposed within the “On musical genre classification via compressive sampling” by Kaichun K. Chang, Jyh- Shing Roger Jang and Costas S. Iliopoulos which was published in 2013, a CS based classifier will be used for musical genre classification. They also claimed to verify the performance of this classifier on a general dataset. Usage of several feature sets were also tried to enhance the performance of classifier. These experiments confirmed that the combination of a CS based classifier along with multiple feature sets can surpass the results and efficiency of all the older approaches.
As always suggested in “Music Genre Classification Using the same Locality Preserving Non-Negative Tensor Factorization and Sparse Representations” by Yannis Panagakis and Constantine Kotropoulos which was published in 2009, a robust framework which makes use of scattered depiction can be used for genre classification. A technique named LPNTF which assimilates the primary spatial structure of the scattered representation in accordance with the music genre into NFT was the main theme used in this framework.

2.1 Merits of Proposed System

In today's digital world, music streaming apps such as Spotify, iTunes, YouTube Music, etc. have become a part of our daily life. This prediction model can be used in these apps to classify the genre of songs. These classified genres can be used later to make a collection of songs pertaining to a specific genre. This provide better user interaction which leads to the success of a music streaming platform

Chapter 3
Functionality/Working of Project

Working of Project:

The practicality of our proposed research can be assessed by following below mentioned approach: -

- Take a sample dataset and convert it into spectrum.
- Label spectrum to a music genre.
- Prepare training and testing datasets
- Implement a classifier model
- predict the music genres for testing dataset

Hence, the accuracy of the model and feasibility of the research can be concluded

Dataset:

GTZAN dataset was utilized to conduct this study.

This dataset primarily consists of audio files pertaining to 10 music genres. There are 100 audio files of each music genre. The duration of each audio file is 30 seconds. The visual depiction of each audio file is also provided in the dataset. One of the approaches to distinguish music genres in this dataset is through neural networks. For this approach to work, first the audio signals has to be transformed into MEL spectrograms. Once these spectrograms are created, then these images can work as an input to neural network. This dataset also consists of 2 csv files which is containing the of audio files. The first csv file contains the value of mean and variance which has been computed over various features of each song (30 seconds duration). The second csv file consists a similar pattern as the first csv file but with one difference. Before calculating the mean and variance for each song, every song has been split into 10 audio files, each of 3 seconds duration. In this research, we would utilize the audio files which are belonging to below mentioned 10 Genre tags only:-

| Blues - 100 |

| Classical - 100|

| Disco – 100 |

| Country – 100 |

| Hip-hop – 100 |

| Jazz – 099 |

| Pop – 100 |

| Rock – 100 |

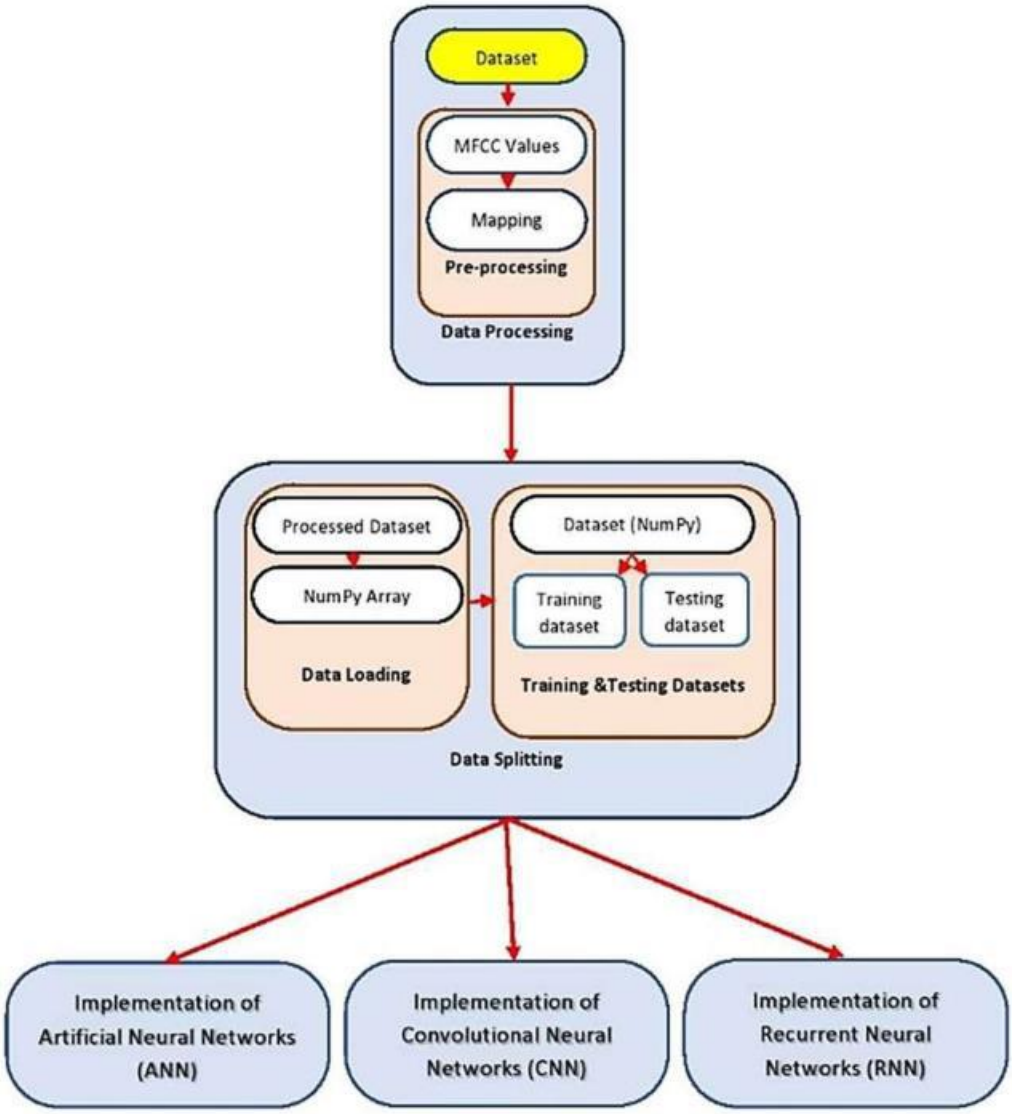
| Metal – 100 |

| Reggae – 100 |

There are two csv files: (1)features_30_sec.csv – size 1.06 MB

(2)features_3_sec.csv – size 10.56 MB

Architecture Diagram:



Methodology:

In this area, the particulars of data pre-processing steps are explained. This section also contains the different approaches to this classification problem

Data Processing :

It provides a per-processed dataset for our Machine learning model. We are using labelled dataset i.e., “GTZAN” by marsyas.info. So, here we are trying to find the Mel-Frequency Cepstrum Coefficients (MFCCs) of our data. MFCC captures timbral or textural aspects of sound. It is a frequency domain feature and works almost like a human auditory system. It resulted in a bunch of coefficient vectors and we can calculate all of these coefficients at each frame so that we can check how the MFCCs are evolving over time. After this we mapped these values with their genre categories.

Spectrogram Generation

Spectrogram is a 2-D pictorial depiction of a signal where the time is always represented on x-axis and frequency is also represented on y-axis. To evaluate the value of a given frequency within a provided time window, a color map is used. During this research, all the audio signals are converted into multiple MEL spectrograms. These spectrograms consist of MEL frequency bins which are represented on y-axis. The below mentioned factors were taken into consideration while generating power spectrogram using STFT: -

- Sampling rate (SRate) = 22050
- Frame or Window Size (fez) = 2048
- Time advance b/w frames = 512
- Freq. Scale: MEL
- No. of MEL: 96
- Frequency(max) = srate/2

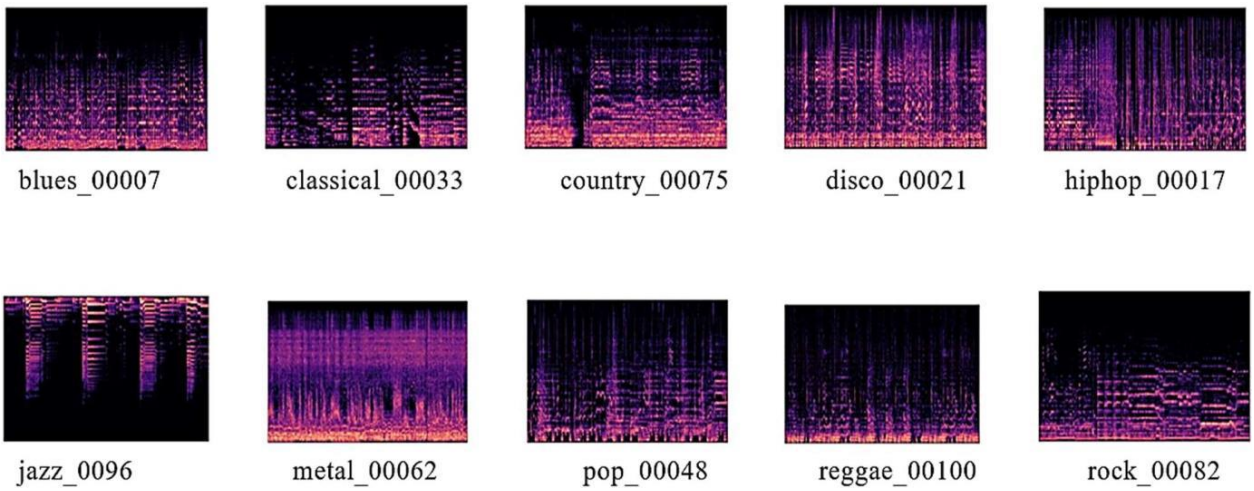


Figure 1: (a) Spectrograms for a sample audio data

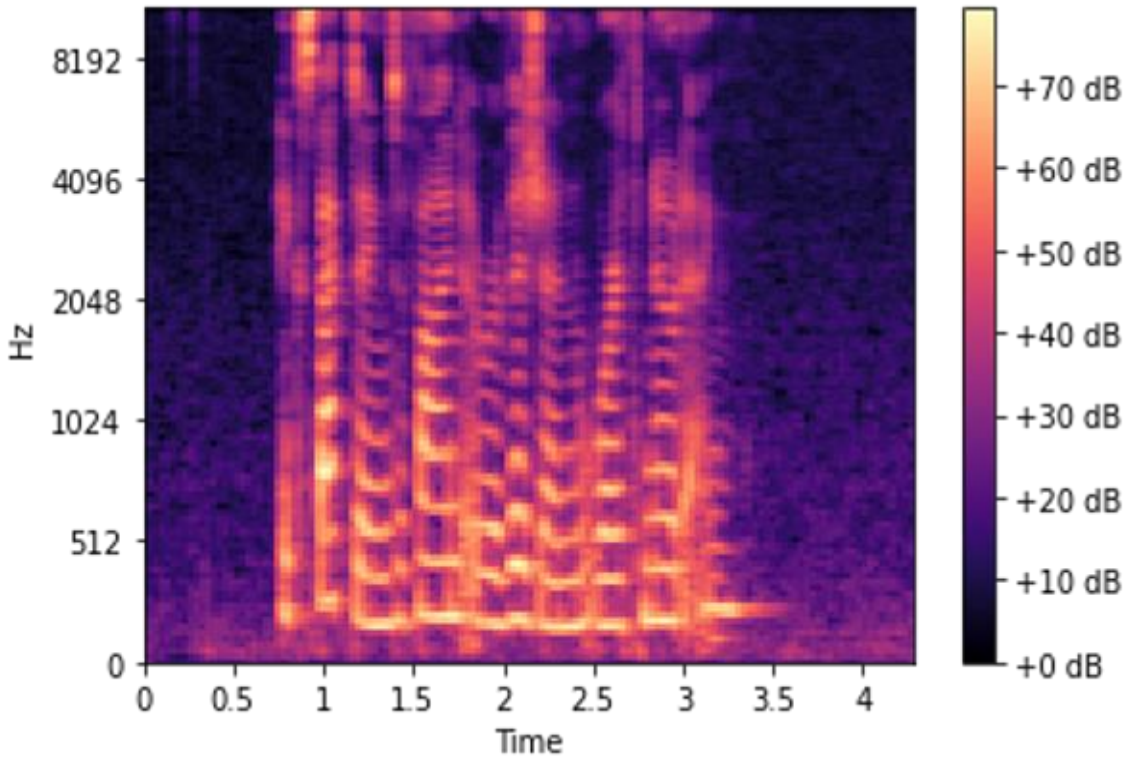


Figure: (b) Mel Spectrogram

```
[37]: stft = librosa.stft(data)
      stft_db = librosa.amplitude_to_db(abs(stft))
      plt.figure(figsize=(14, 6))
      librosa.display.specshow(stft_db, sr=sr, x_axis='time', y_axis='hz')
      plt.colorbar()
```

```
[37]: <matplotlib.colorbar.Colorbar at 0x7fa44f047160>
```

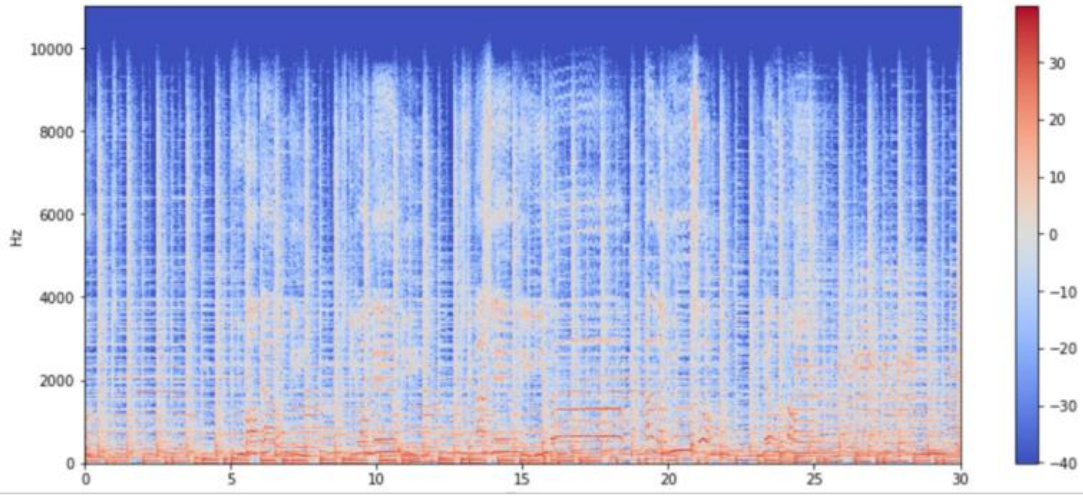


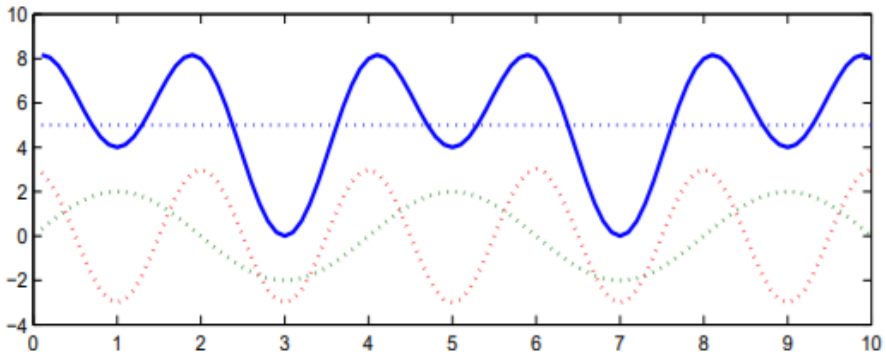
Figure 2: The vertical axis represents frequencies (from 0 to 10kHz), and the horizontal axis represents the time of the clip.

MFCCs (Mel-Frequency Cepstrum Coefficients)

The MFCC highlight extraction procedure essentially include windowing the sign, applying the DFT (Discrete Fourier Transform), taking the log of the magnitude, and afterward distorting the frequencies on a Mel scale, followed by applying the opposite DCT (Discrete cosine Transform).

DFT (Discrete Fourier transform)

The DFT changes over a limited succession of similarly dispersed sample of a function into an equivalent length grouping of similarly separated examples of the discrete-time Fourier Transform (DTFT), which is a complex-esteemed capacity of frequency.



Example signal for DFT

DCT (Discrete cosine transform)

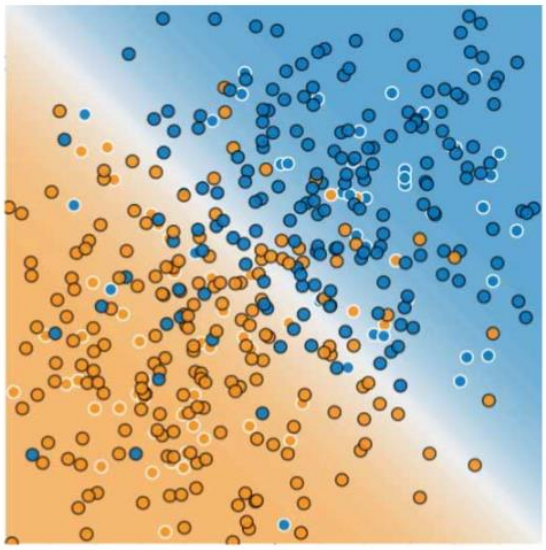
A discrete cosine transform (DCT) is the sum of cosine functions expressed form sequence of data points oscillating at different frequencies.

Figure: showing eight different filters applied to a test image (top left) by multiplying its DCT spectrum (top right) with each filter.

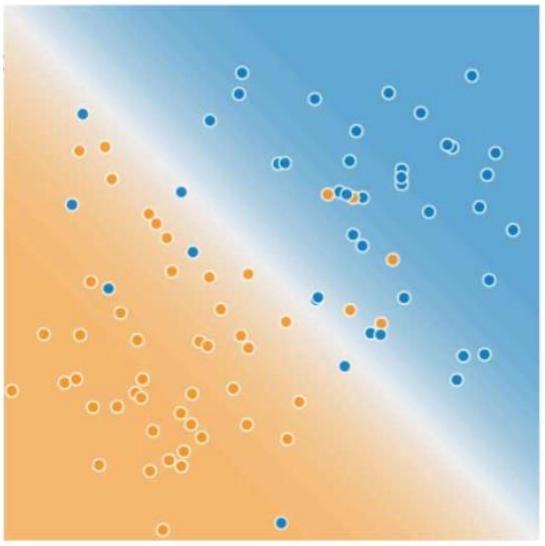
s

Data Splitting

In this step, the pre-processed dataset is split up into 2 parts- training dataset and testing dataset. Firstly, we had to load the data into the model so we convert it into NumPy array. This provides ease to access the data and some extra details too. Then we split the dataset, it should be randomized data for better model training. For this we had to import some modules from Scikit Learn (or sklearn) to split the dataset. It automatically took random data form dataset and create training and testing dataset



Training Data



Test Data

NumPy

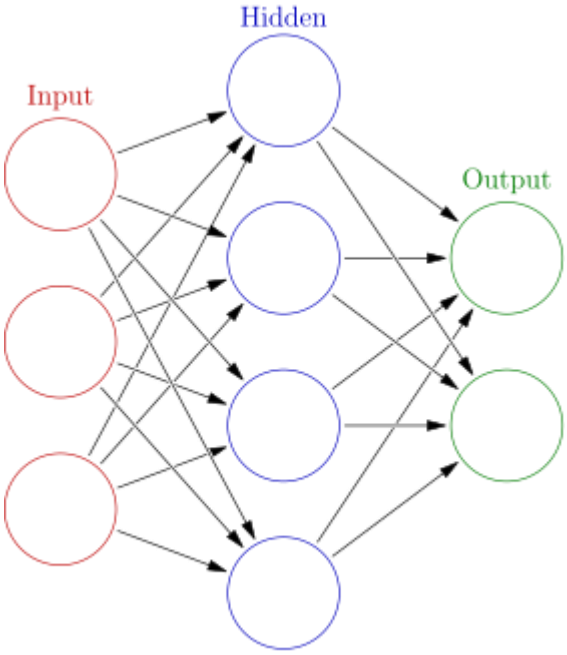
NumPy is a library for the Python programming language, adding support for enormous, multi-dimensional arrays and networks, alongside a huge assortment of high- level numerical functions to work on these arrays.

Scikit-learn

Scikit-learn (formerly scikits.learn and also known as sklearn) is a free software machine learning library for the Python programming language. It consists of several models/algorithms for both supervised as well as unsupervised training along with DBSCAN algorithm. It is devised to interact with Python libraries.

Artificial Neural Network (ANN)

As we implemented multi-class classification which is opposite to binary classification so, we select the multi-layer perceptron which is a simple neural network. But the question arises why do we need a neural network? Isn't an artificial neuron good enough? So, the answer to this question is that if we want to work on quite complex problems like real-world problems then we need to scale up our network. A bigger network of neurons is required. Since a single neuron is highly efficient in dealing with simple problems which are associated with linearity but in this situation, we are dealing with more complex problem which is associated to non- linearity. Therefore, we need a network of neurons which work together. These networks can produce highly nonlinear functions which can help us in solving highly nonlinear problems.



Neural Architecture

(a) Sequence Model

Sequence models are the machine learning models that info or result Sequence of information. Sequence information incorporates text transfers, sound bites, video cuts, time-series information and so on So here we used an input layer, three hidden layers and an output layer. The Input layer that we used here is flatten. Hidden layers are the simple dense layers.

(b) Activation function

In hidden layer, we generally use the Sigmoid function but this time we used a new type of activation function that's called relu. Relu function is highly effective in deep learning. It empowers us to prepare the organization way quicker than the function. As a result, we get better convergence of the network. This is because relu function reduces the probabilities of vanishing gradient. In output layer, the activation function used is soft max.

$ReLU(h) = \{0, if\ h < 0$

$h, if\ h \geq 0$

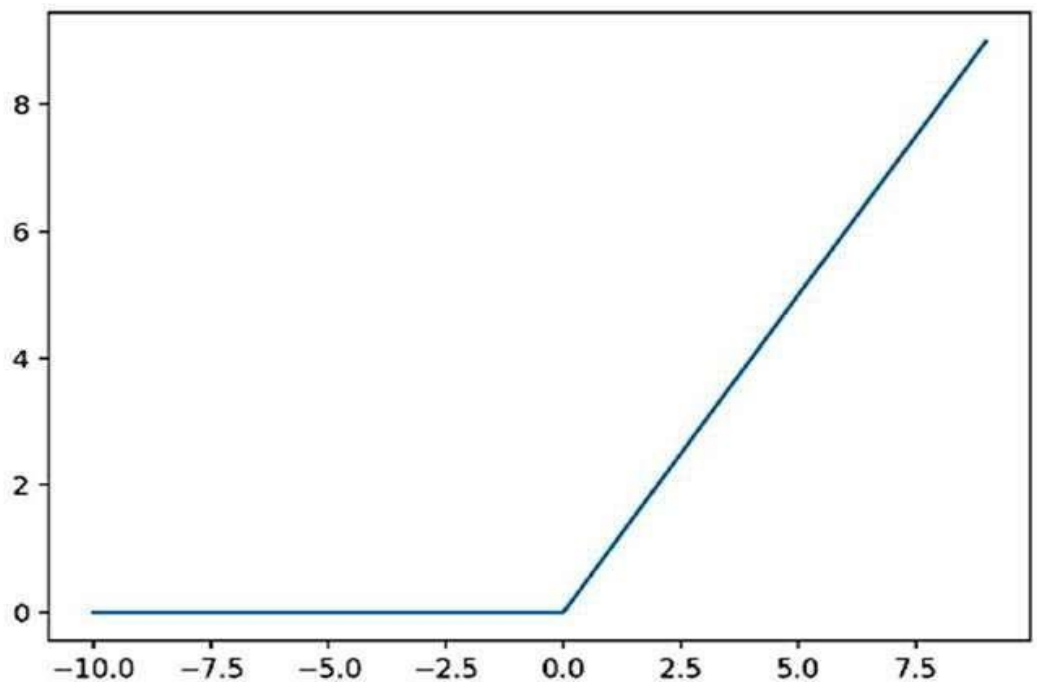
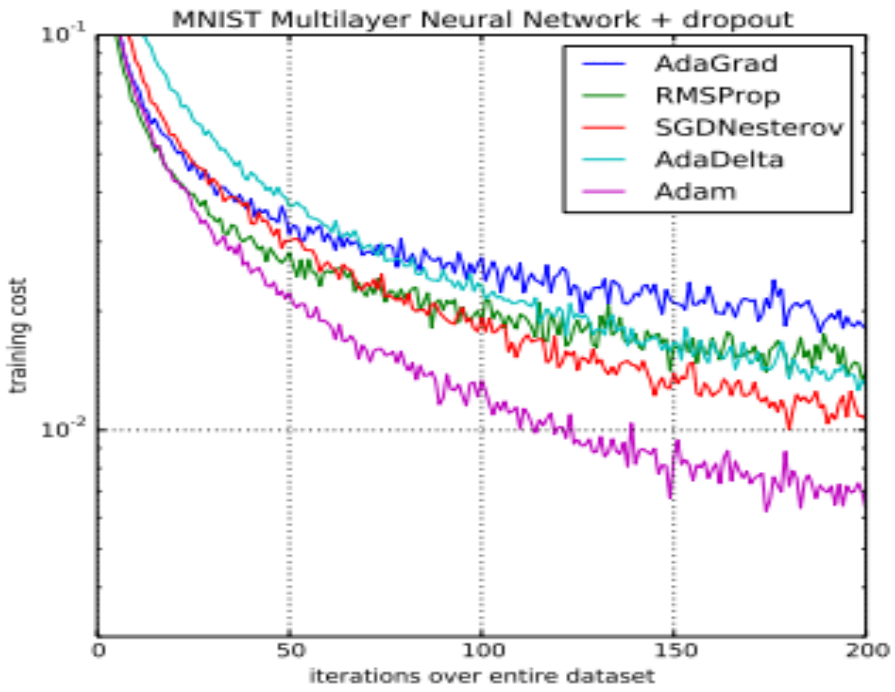


Figure 3: Rectified Linear (ReLU) activation function

Adam Optimizer

Adam is an optimizer that is basically like a variation of a stochastic slope plunge for preparing deep learning models. The best properties of AdaGrad and RMSProp algorithm are joined by this streamlining agent to give another algorithm. This new algorithm can deal with meager slopes on loud issues.One more advantage of using this optimizer is its ease of configuration as most of the default configuration parameters work on most problems.



Comparison of Adam to Other Optimization Algorithms Training a Multilayer Perceptron. Taken from Adam: A Method for Stochastic Optimization, 2015.

Batching

Batching is the process of identifying the number of samples to be used from the training dataset. These samples then would be used to calculate gradient. There are different types of batching which is used to retrain a keras network.

- **Stochastic Gradient Descent**: To calculate the gradient in this method, only one sample segment of the dataset is considered. So, a feed-forward is performed which is followed by a back propagation and hence, the gradient is calculated. After that the weights are updated directly. This method is quite quick to perform but it gives highly inaccurate results. It is due to the presence of huge amount of noise which is equivalent to a scenario where batch size is equal to one.
- **Full batching**: To calculate the gradient in this method, the complete training dataset is used. So, the full training dataset is gone through the model and the angle is determined. After that the loads are refreshed all in all training set. The biggest disadvantage of this method is that usually datasets are really huge in size which makes this complete process super slow and highly memory intensive. It's almost impractical. The largest advantage of this method is that results are

highly accurate because we are calculating the gradient on all the samples within the training set.

• **Mini- batch:** To calculate the gradient in this method, a subset of training set is used. This subset is a set of 16 -128 samples which are selected from the training dataset. Hence, this method is called a mini- batch. Once the subset is finalized, the gradient is calculated for all the chosen samples and weights are updated afterwards. The no. of samples to be selected in the subset depends on the type of problem that is being dealt with. This method is basically a middle ground of both the above-mentioned methods. So, the process is relatively quick and less memory intensive than Full Batch method. Also, the results are quite accurate when compared to Stochastic Gradient method. Hence, we use this method of batching.

Overfitting

The result that we achieved on training dataset from neural network was highly accurate i.e. 97% whereas the same network didn't give such accurate results when tested with full dataset. There was a huge difference of approximately 40%. Hence, we observed that it's a case of overfitting. Basically, it implies that the model was performing very well on the training set but it is facing issues with data it had never seen before. Here, we are using 2 methods to remove overfitting which are as follows: -

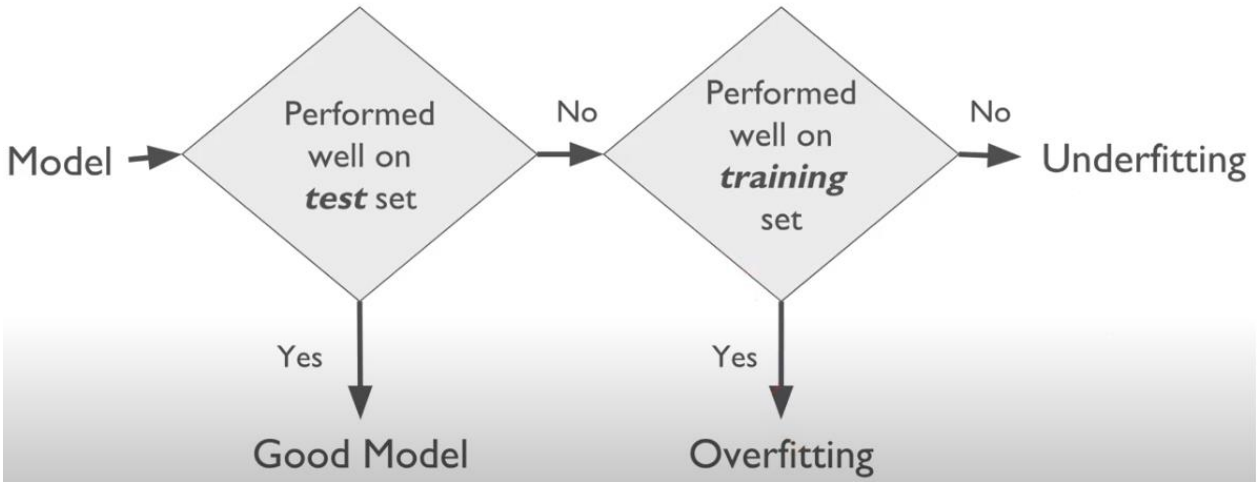


Figure:4 Overfitting

Dropout

Dropout is a technique that enables us to randomly drop neurons while training the model which in turn increases the network robustness. For example, we have our first batch of data and we decide to randomly drop certain neurons. In figure-3, we dropped two neurons which are shown in grey So, all the connections won't work for these neurons and hence, the training just happens with the remaining parts of the network.

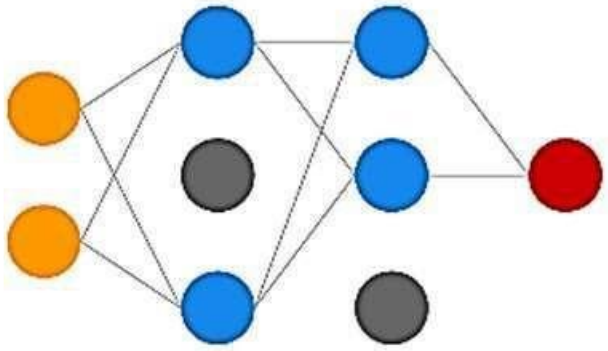


Figure 5: Dropping of neurons

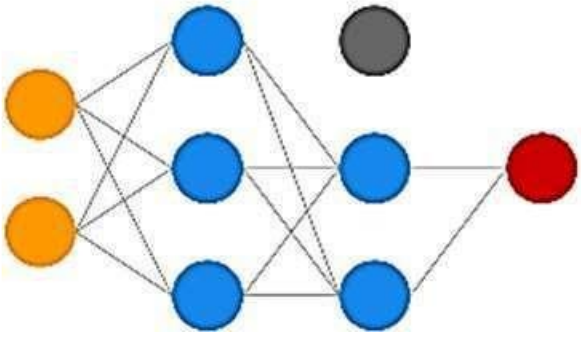


Figure 6: Change in a dropped neuron

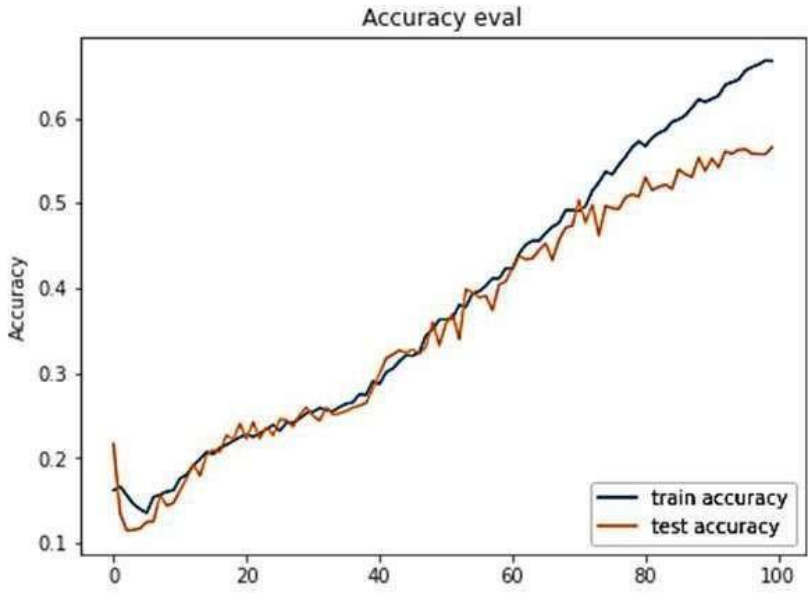
Similarly, with the second batch of data, wecan just change the dropped neurons andthis would be done randomly. We have a certain probability of dropping neurons in a layer. As shown in figure-4, we drop this grey neuron in this second hidden layer and we restore the previous neurons.

Hence, we are increasing the network Robustness. Since the network can't rely too much on specific neurons, so all the neurons have to take some responsibility of the prediction process. It is similar to reshaping of the network which then redistribute responsibilities to all of the neurons so that none of them is indispensable

Regularisation

It's a technique that adds penalty to the error function. The main idea here is to punish large weights. The larger the weights, the higher would be the penalty given to the error function. There are two types of regularization that are generally used in deep learning. These are L1 and L2 regularization. In L2 regularization, we minimize the squared value of the of the weights as shown in formula. As a consequence, this method is way less robust to outliers but the advantage of this method is that it can learn quite complex.

$E(p, y) = 1\ (p - y)2 + \boldsymbol{\lambda}\ \Sigma\ \boldsymbol{W2\ i}$



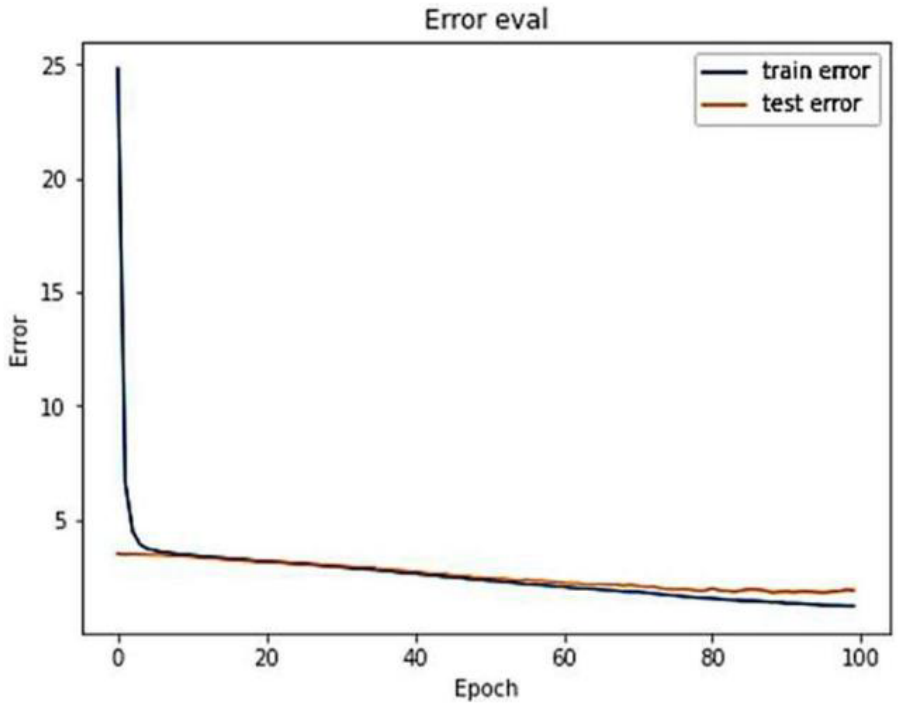


Figure 7: ANN Learning Curve

Convolutional Neural Networks (CNN)

As per Methodology, anyone can recognize that different genres have different characteristic patterns. These patterns are visible in the spectrograms of these audio signals. Therefore, spectrograms are basically images which work as input to a CNN.

CNN are the high level sort of neural network and they are mainly used for processing images and all the time we've found out that they perform well with images than the equivalents. For example, multi-layer perception architecture. The good thing about this architecture is that they require very less parameters than dense layers.

Image data is like structured data and pixels are not randomly positioned. There are certain emergent structures like edges, shapes, invariance to interpretation and scale invariance. CNN is trying to emulate human vision system. We extract basic features and CNN try to learn those different types of features. All of this process relies on a couple of components i.e., convolution and pooling.

Convolution

Convolution is used for many things like calculating derivatives, detect edges, apply blurs etc. and all this is done using a "convolution kernel". A convolution kernel is a very small matrix and in this matrix, each cell has a number and also an anchor point.

The anchor point is used to know the position of the kernel with respect to the image. It starts at the top left corner of the image and moves on each pixel sequentially. Kernel overlaps few pixels at each position on the image. Each pixel which is overlapped is multiplied and then added. And the sum is set as the value of the current position.

At this stage, a matrix filter (Kernel) is slide over the input image. On the image, kernel is placed first. Then we compute an Hadamard product of the kernel and the overspread segment of the image. The summation of these values gives us a feature value. During the training, we use many such kernels which are learned via backpropagation.

0	**-2**	**0**
-1	**0** anchor	**-1**
0	**-2**	**0**

Convolution is the process in which each element of the image is added to its local neighbors, and then it is weighted by the kernel. It is related to a form of mathematical convolution.

Suppose, there are two 3x3 matrices, one is kernel and another one is an image piece. In convolution, rows and columns of the kernel are flipped and then they are multiplied and then summing is performed. Elements which are present in the centre of matrix i.e. in [2,2] of the image will be weighted combination of the image matrix and the weights will be given by the kernel. Similarly, all the other elements of the matrix will be weighted and then weights will be computed.

Kernal

Kernel or a channel is a component identifier which various kernels identify various elements. For example,

In figure - 6, these kernels used to identify oblique lines and vertical lines.

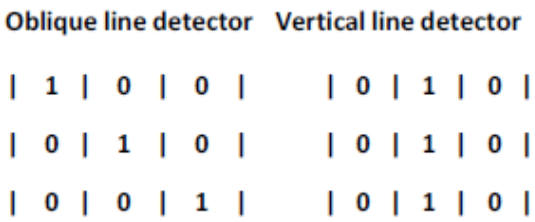


Figure 8: Kernel for Detecting Different Lines

We have different types of architectural decision which helps to decide the type of convolution. Since, we were using spectrogram as images and those are colored images. Hence, ‘depth’ is the key feature. In grayscale image the depth is equal to one but if you have a colored image i.e., in RGB each pixel has three qualities one for the red, one for the green and one for the blue colour. In terms of kernel,it is divided into three parts or three grids where one grid is for red, one for green and one for the blue channel. As these grids are independent, we are getting three-dimensional array where first two dimensions represents the width and the stature of the of the kernal and the third dimension address the 'depth'.

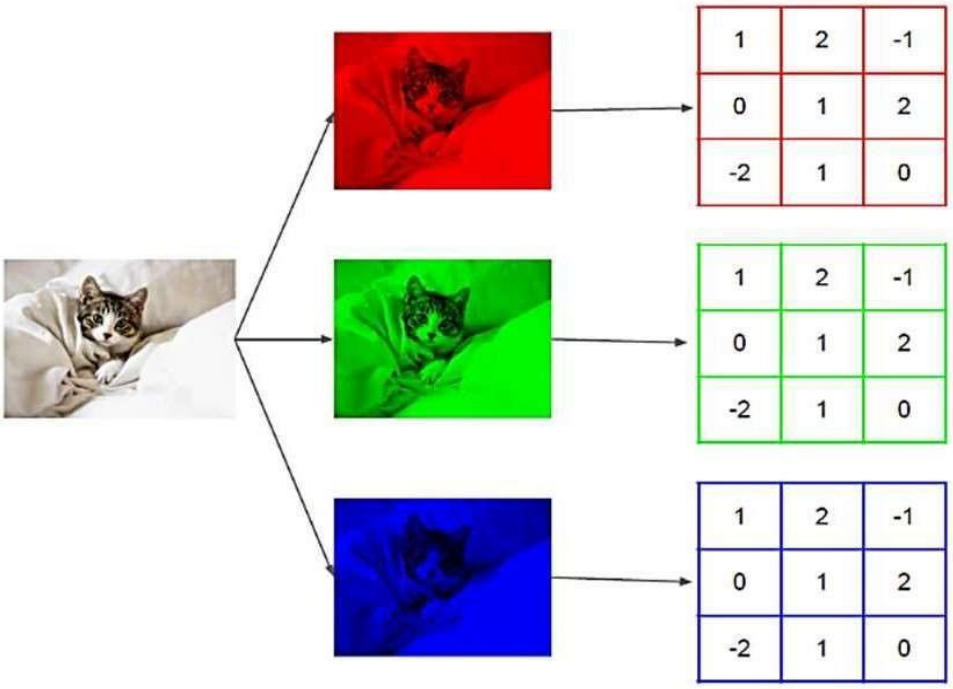


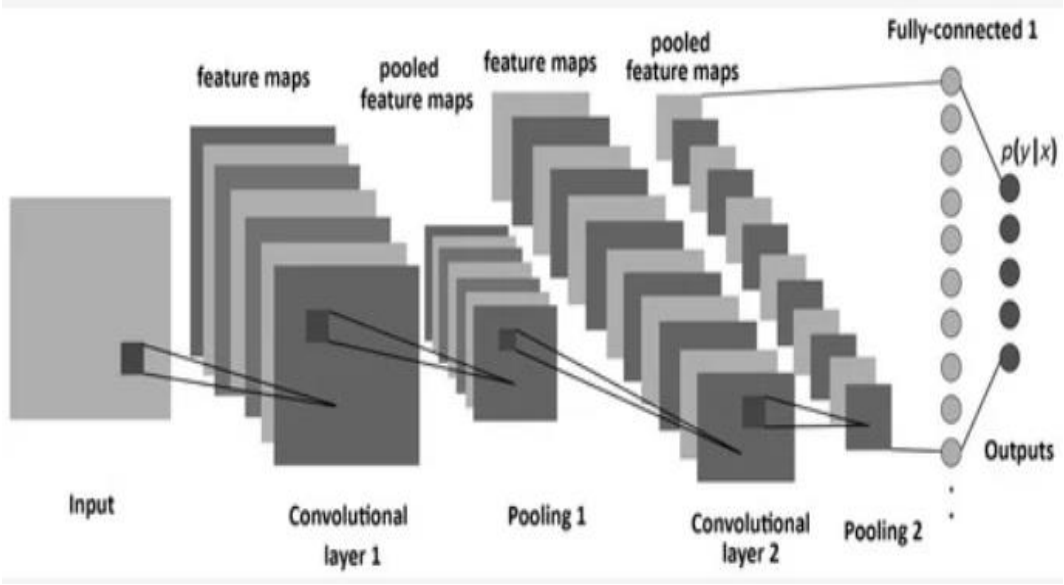
Figure 9: Depth of a RGB image

Feed Forward Network

A CNN is a feed-forward network, that is, input examples are fed to the network and transformed into an output; with supervised learning, the output would be a label, a name applied to the input. That is, they map raw data to categories, recognizing patterns that may signal, for example, that an

input image should be labeled “folk” or “experimental”. A feedforward network is trained on labeled images until it minimizes the error it makes when guessing their categories. With the trained set of parameters (or weights, collectively known as a model), the network sallies forth to categorize

data it has never seen. A trained feedforward network can be exposed to any random collection of photographs, and the first photograph it is exposed to will not necessarily alter how it classifies the second. Seeing spectrogram of a folk song will not lead the net to perceive a spectrogram of an experimental song next. That is, a feedforward network has no notion of order in time, and the only input it considers is the current example it has been exposed to. Feedforward networks are amnesiacs regarding their recent past; they remember nostalgically only the formative moments of training.



Basic CNN model training:

After preprocessing the data, we create our first deep learning model. We construct a Convolution Neural Network model with required input and out units. The final architecture of our CNN model is shown in Figure 04. We use only spectrogram data for the training and testing.

```
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 (MaxPooling2 (None, 64, 64, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 64, 64, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 32, 32, 64)        0
_________________________________________________________________
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
_________________________________________________________________
```

Figure 10: We train our CNN model for 500 epochs with Adam optimizer at a learning rate of 0.0001. We use categorical cross-entropy as the loss function

Pooling:

This is the process of dimension reduction into feature map by convolution steps. This process is called as down sampling. It works in similar manner as convolutional i.e.,overlay a grid on top of an image. Generally, for deep learning max pooling is used with 2x2 matrix. Out of the 4 elements inside the matrix, the maximum value is picked up. We keep moving this window across the feature map with a predefined stride.

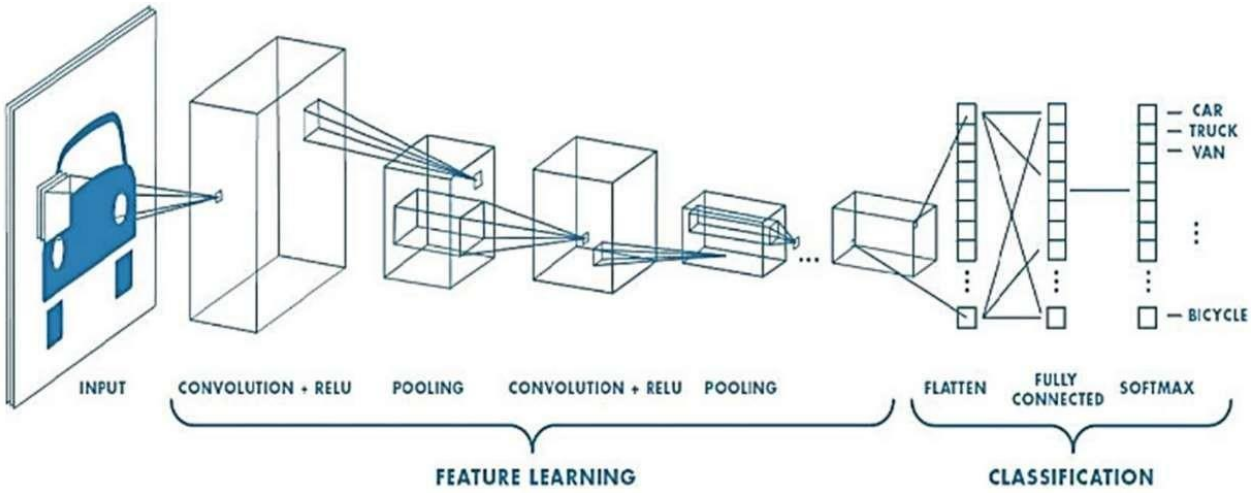
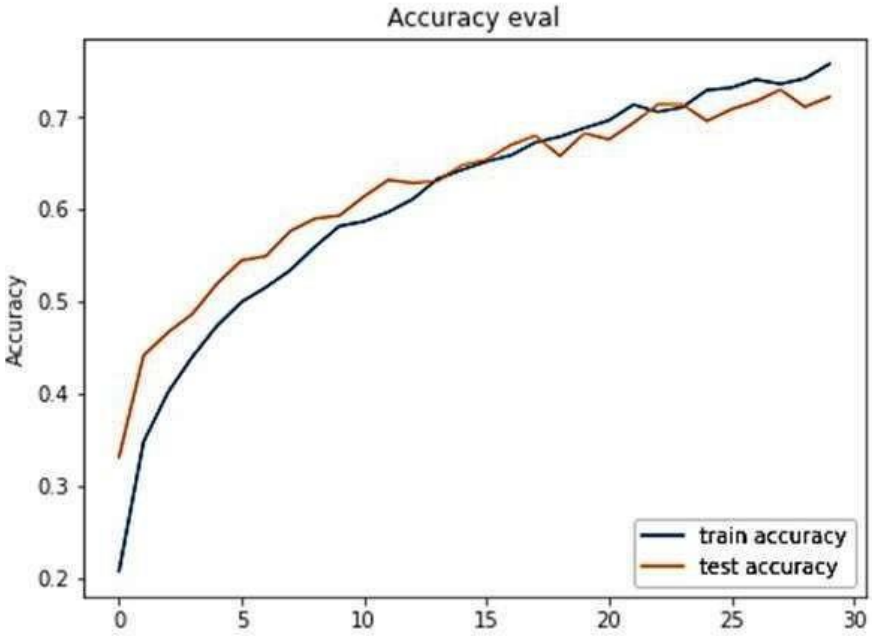


Figure 8: Convolutional Neural Networks (CNN) architecture

Implementation of convolution and pooling

In case of spectrograms, the time and frequency can be considered as x and y indexes for the pixels of an image. Also, amplitude can be considered as the value associated to each pixel. This concept is basically comparing a spectrogram with a 2-D array. For example, we have 13 MFCCs, 512 samples as Hop length and51200 samples in an audio file then the expected data shape which would be in our CNN is 100x13x1. Here, 100 denotes the time windows at which we take 13 values. 13 denotes the no. of MFCCs and 1 denotes the depth.



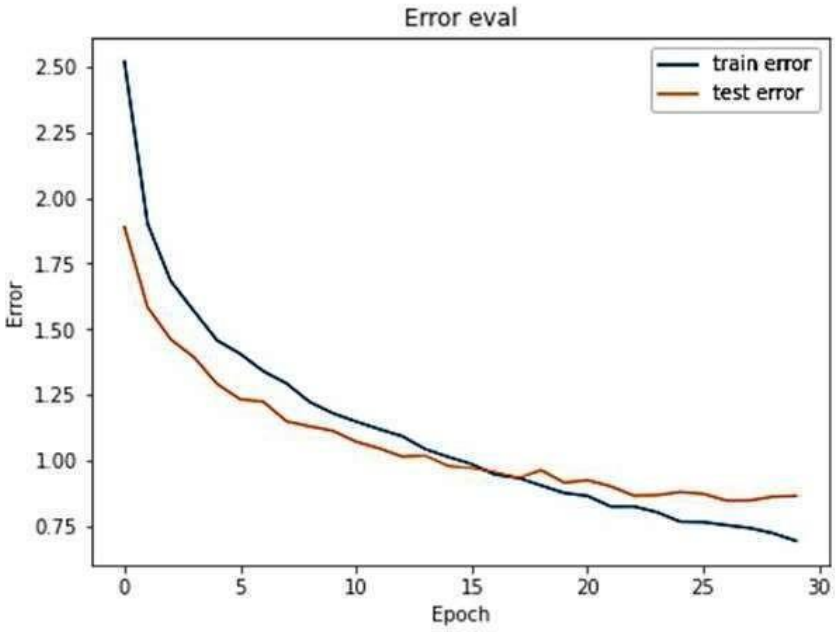


Figure 12: CNN Learning Curve

Testing the models

After training our models, we test each model on the 40% test data. We calculate precision, recall, and F-score for each music genre (class). Our dataset is balanced; therefore, the macro average and weighted average of precision, recall, and F-score are the same.

(A)Basic CNN model

Figure 13 presents the results of our CNN model on the test data. CNN model was able to classify “classical” genre music with the highest F1-score. CNN performed worst for “Rock” and “reggae” genre music. Figure 14 shows the confusion matrix of the CNN model on the test data.

	precision	recall	f1-score	support
blues	0.46	0.55	0.50	40
classical	1.00	0.80	0.89	40
country	0.42	0.42	0.42	40
disco	0.40	0.45	0.42	40
hiphop	0.41	0.60	0.49	40
jazz	0.57	0.50	0.53	40
metal	0.81	0.72	0.76	40
pop	0.72	0.53	0.61	40
reggae	0.37	0.38	0.37	40
rock	0.28	0.25	0.26	40
accuracy			0.52	400
macro avg	0.54	0.52	0.53	400
weighted avg	0.54	0.52	0.53	400

Figure 13

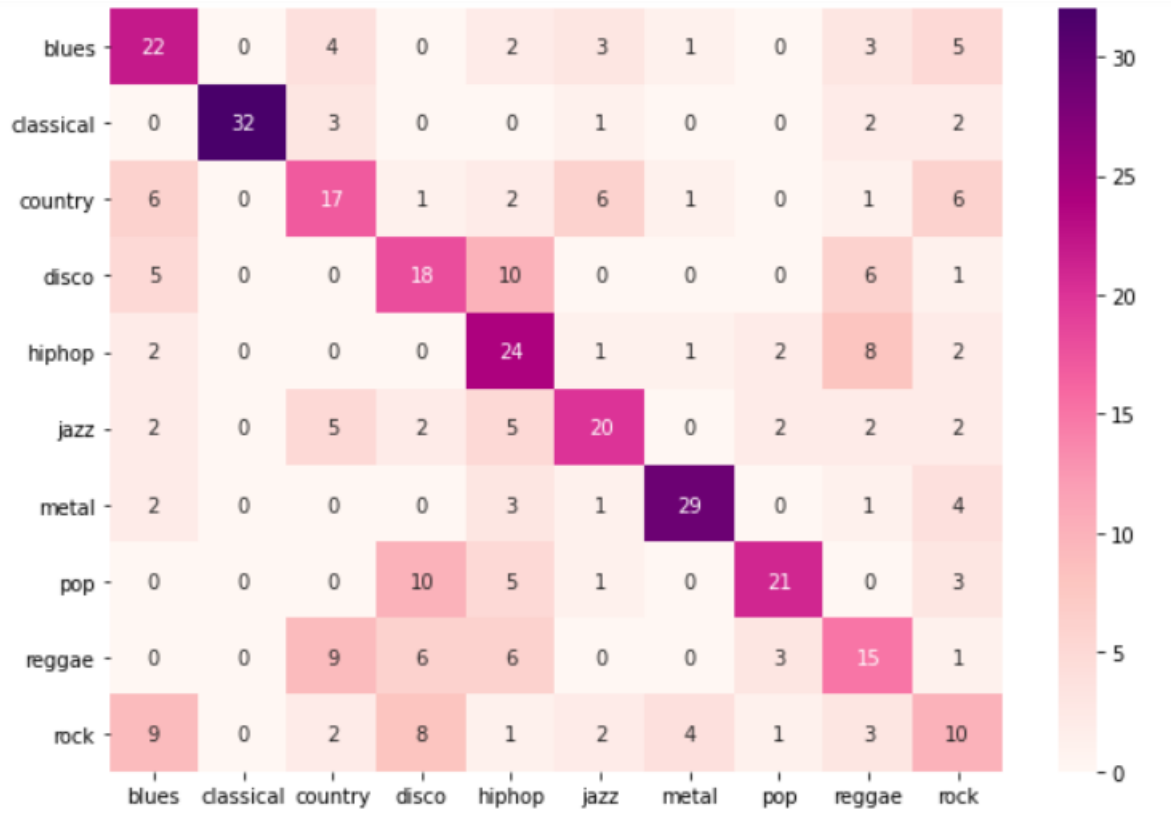


Figure 14

(B)Transfer learning based model

We used the transfer learning technique to improve the performance of genre classification. Figure A presents the results of the transfer learning-based model on test data. F1-score for “hiphop”, “jazz”, and “pop” genres increased due to transfer learning. If we look at overall results, we have achieved only a minor improvement after applying transfer learning. Figure B shows the confusion matrix for the transfer learning model on the test data.

	precision	recall	f1-score	support
blues	0.43	0.57	0.49	40
classical	0.90	0.88	0.89	40
country	0.43	0.33	0.37	40
disco	0.46	0.45	0.46	40
hiphop	0.43	0.72	0.54	40
jazz	0.81	0.65	0.72	40
metal	0.67	0.60	0.63	40
pop	0.84	0.65	0.73	40
reggae	0.35	0.33	0.34	40
rock	0.26	0.23	0.24	40
accuracy			0.54	400
macro avg	0.56	0.54	0.54	400
weighted avg	0.56	0.54	0.54	400

Figure: A

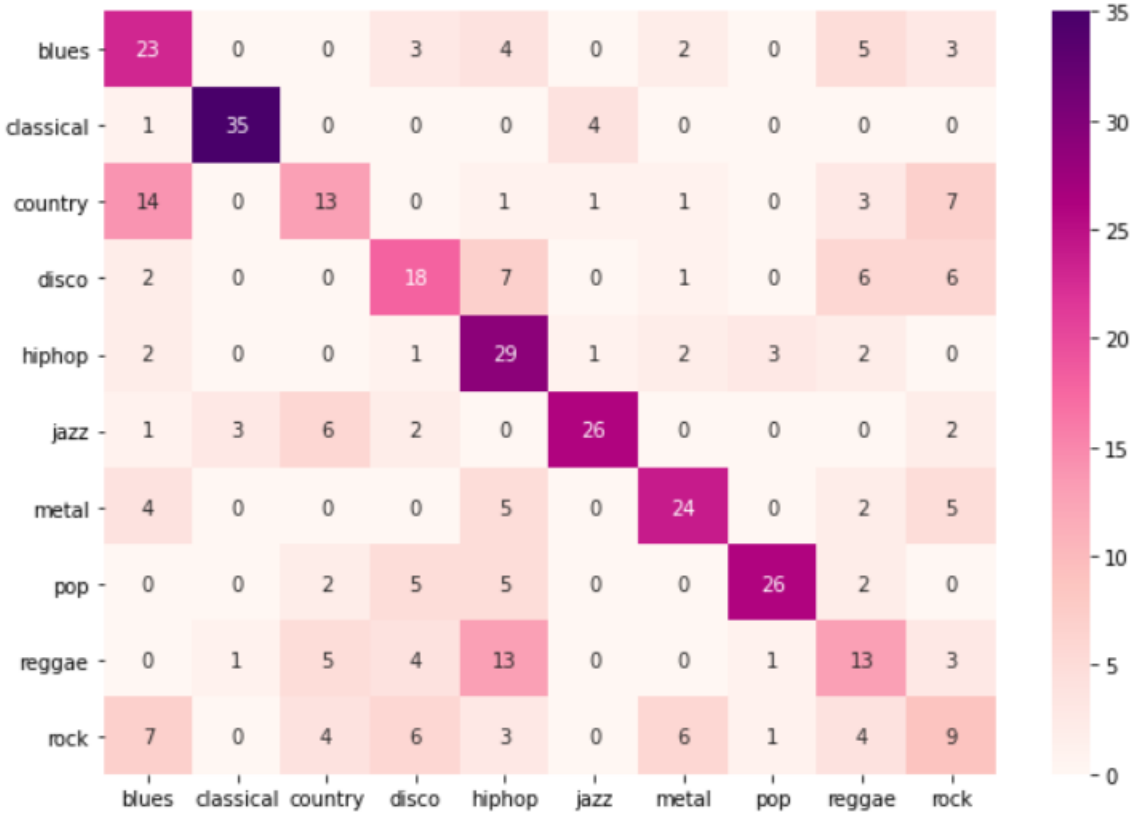


Figure: B shows the confusion matrix for the transfer learning model on the test data.

Recurrent Neural Networks (RNN)

RNN is a process which is used for the dataset where order is extremely important. Some examples of such dataset are time series or measures of different values that are taken at fixed time intervals. The dataset can have variable amount of words/length like musical notes. Hence, this method is ready to handle consecutive information so that each data point is processed in context. It is an ideal method for audio processing.

As we know that audio files are in the format of waveform. So, we can consider these as a univariate time series. It means that we have just one worth or one measure that is taken at every stretch. Contrary to this dataset, there are another type of dataset such as MFCCs which are multivariate time series. Multivariate time series are the ones where we take more measurements for each interval and the relative dimension is given by the overall number of intervals. For a time series dataset, we take input data points one at a time. Our goal is to predict next step on the basis of previous data points which helps the model in understanding context.

In RNN, recurrent layer is the layer that is able to process sequential data in a sequential way. The cell is the one that is responsible for processing this sequential data at each step. We give input data that's represented by this Xt then the cell does some processing and it gives output as Yt and Ht. Ht is called as a state vector or a hidden state which keeps memory of the cell at a certain point in time whereas Yt is the actual output. The whole point of the recurrence here is given by the fact that Ht or the state vector is going to be reused at a later point of time at the next step. In this way, we can have information about the context.

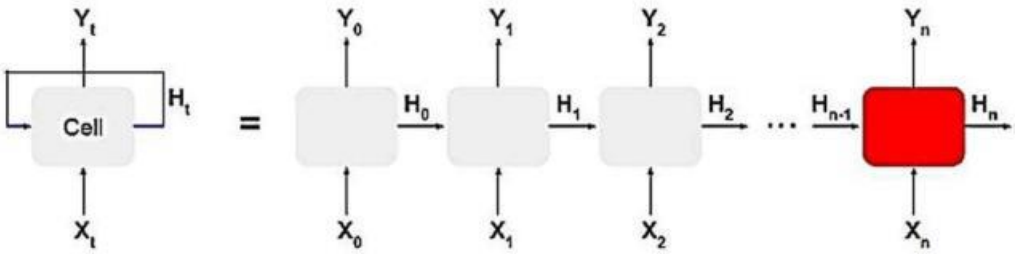


Figure 15: Unrolling a recurrent layer

Memory cell is a very simple neural network and here we use dense layer and the input is given by the combination of the state vector and the input data. The state vector is used at the previous timestamp and the input data is used at the current timestamp. The activation function that we use is the hyperbolic tangent (tan H).

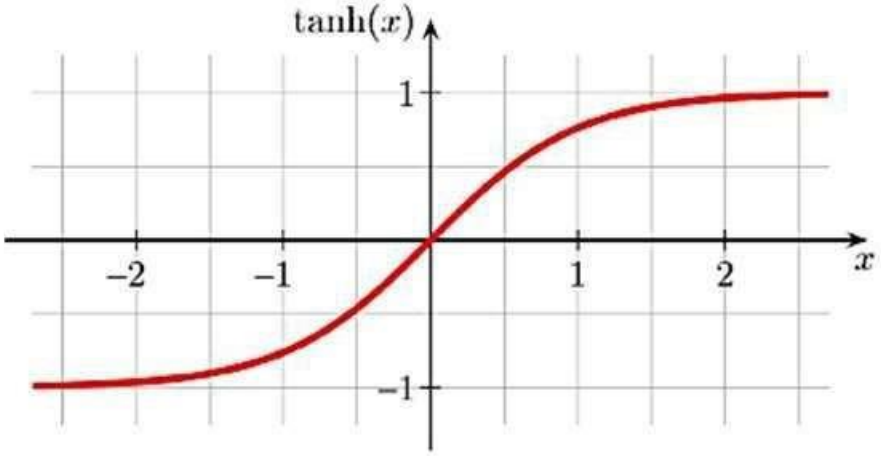


Figure 16: Hyperbolic Tangent (tan H) activation function

We could have used other activation functions but it turns out that the training with RNN is quite difficult because of anumber of reasons. Mainly vanishing gradients where the gradients tend to disappear and at the same time, we have the issue of exploding gradients where the gradients to grow bigger. If we use relu then CNN's revenue is not bounded but in RNN relu can explode whereas tan H constrains the values between -1 and 1. To train the network, we back propagate the error through time. So, RNN is unrolled and treated as feedforward network.

However, there are not many issues with simple RNN. One of the issue is that they don't have a long-term memory. Along these lines, the network can't utilize data from . Likewise, one of the issues is that they can't learn designs with long conditions. To keep away from these issues, various variations have been conceived and a particular one which is extremely fruitful is called long short-term memory network (LSTM)

Long short-term memory network (LSTM).

As a solution to the shortcomings of normal RNNs, long shorts and memory networks was introduced. LSTM are an extraordinary kind recurrence neural networks where we have memory cells that enable us to learn long- term patterns. It’s quite effective in detecting pattern which is up to 100 steps but it struggled with 100s/1000s of steps. In RNNs, the memory cell is a simple dense layer with hyperbolic tangent (tanh) as an activation function but in LSTM the cell structure has been modified. The standard

formulation of a single LSTM cell can be given by following equations:

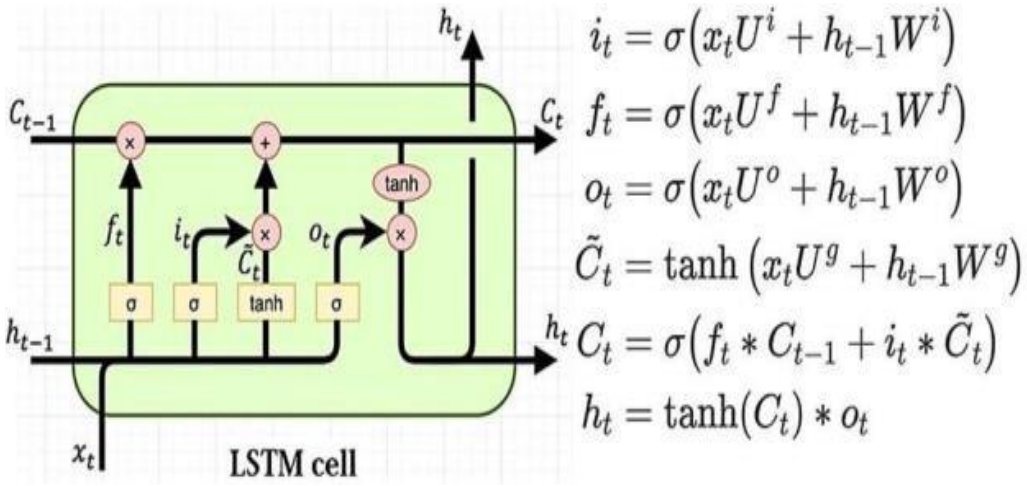


Figure 17: LSTM cell and equations

where i, f, o, C, ˜C are the input gate, forget gate, output gate, memory cell content, and new memory cell content, respectively. σ is the sigmoid function, tanh is the hyperbolic tangent function. The sigmoid function is used to form three gates in the memory cell. On the other hand, the tanh function is used to scale up the output of a particular memory cell. The sigmoid function ranks the output between zero and one. Hence, the values which are closer to zero will be forgotten in the cell state whereas the values which are closer to one will be kept. The tanh function is used to scale up the output of a particular memory cell.

LSTM Architecture

At a high-level LSTM works very much like an RNN cell. Here is the internal functioning of the LSTM network. The LSTM consists of three parts, as shown in the image below and each part performs an individual function

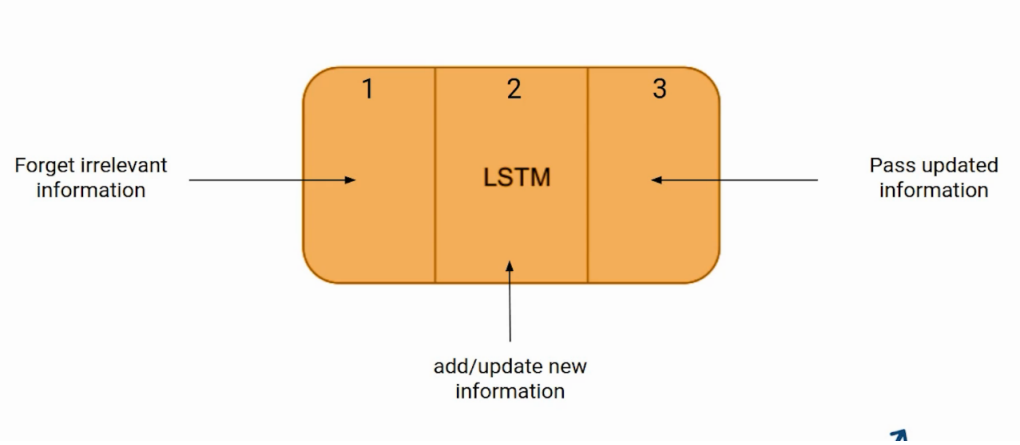
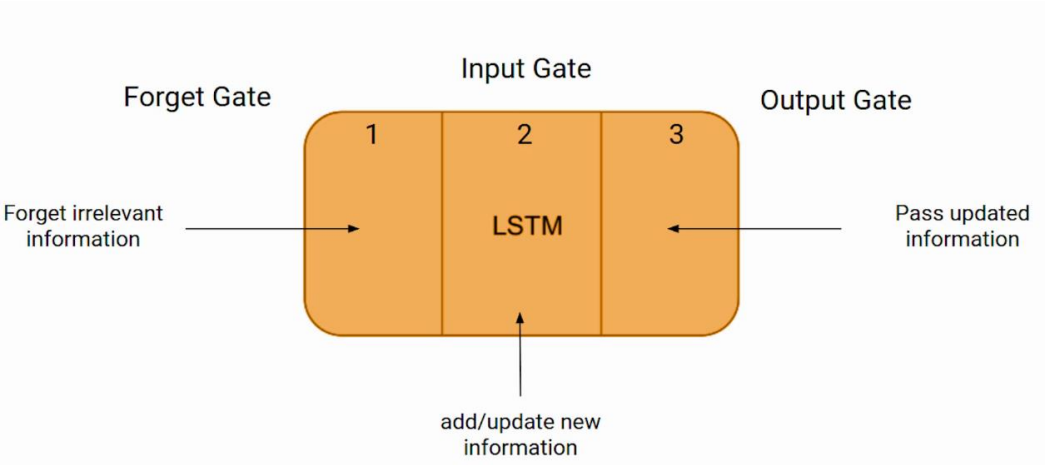


Figure:18: LSTM architecture

The first part chooses whether the information coming from the previous timestamp is to be remembered or is irrelevant and can be forgotten. In the second part, the cell tries to learn new information from the input to this cell. At last, in the third part, the cell passes the updated information from the current timestamp to the next timestamp.

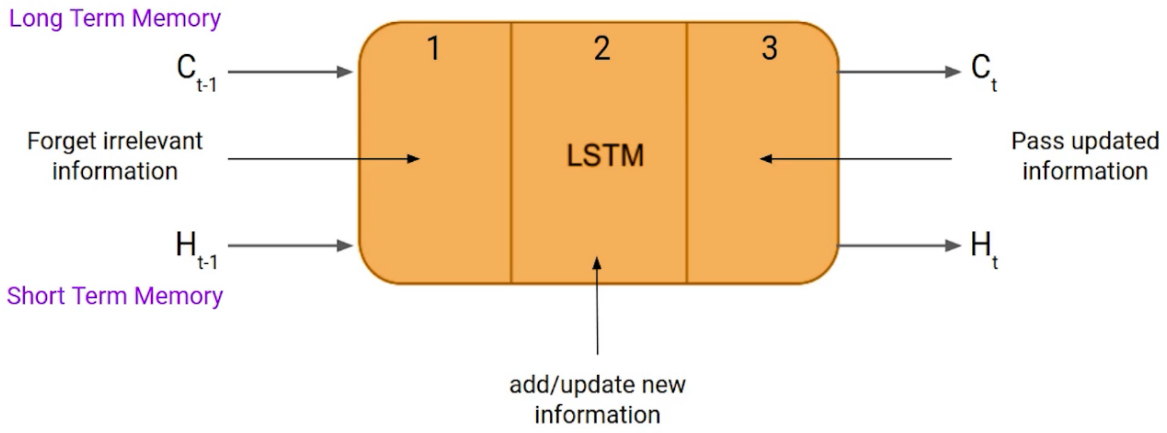
These three parts of an LSTM cell are known as gates. The first part is called Forget gate, the second part is known as the Input gate and the last one is the Output gate.



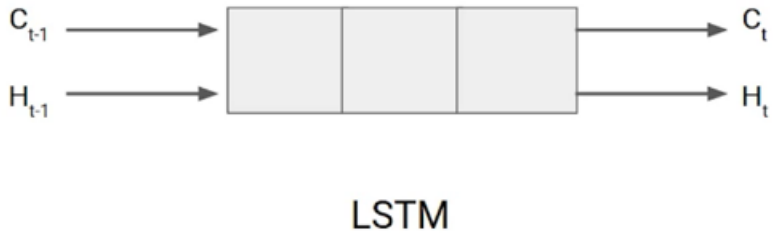
Just like a simple RNN, an LSTM also has a hidden state where H(t-1) represents the hidden state of the previous timestamp and Ht is the hidden state of the current timestamp. In addition to that LSTM also have a cell state represented by C(t-1)

and C(t) for previous and current timestamp respectively.

Here the hidden state is known as Short term memory and the cell state is known as Long term memory. Refer to the following image.



It is interesting to note that the cell state carries the information along with all the timestamps.



Let's take an example to understand how LSTM works. Here we have two sentences separated by a full stop. The first sentence is "Naveen is a nice person" and the second sentence is "Deepak, on the Other hand, is evil". It is very clear, in the first sentence we are talking about Naveen and as soon as we encounter the full stop(.) we started talking about Deepak.

As we move from the first sentence to the second sentence, our network should realize that we are no more talking about Naveen. Now our subject is Deepak. Here, the Forget gate of the network allows it to forget about it. Let's understand the roles played by these gates in LSTM architecture.

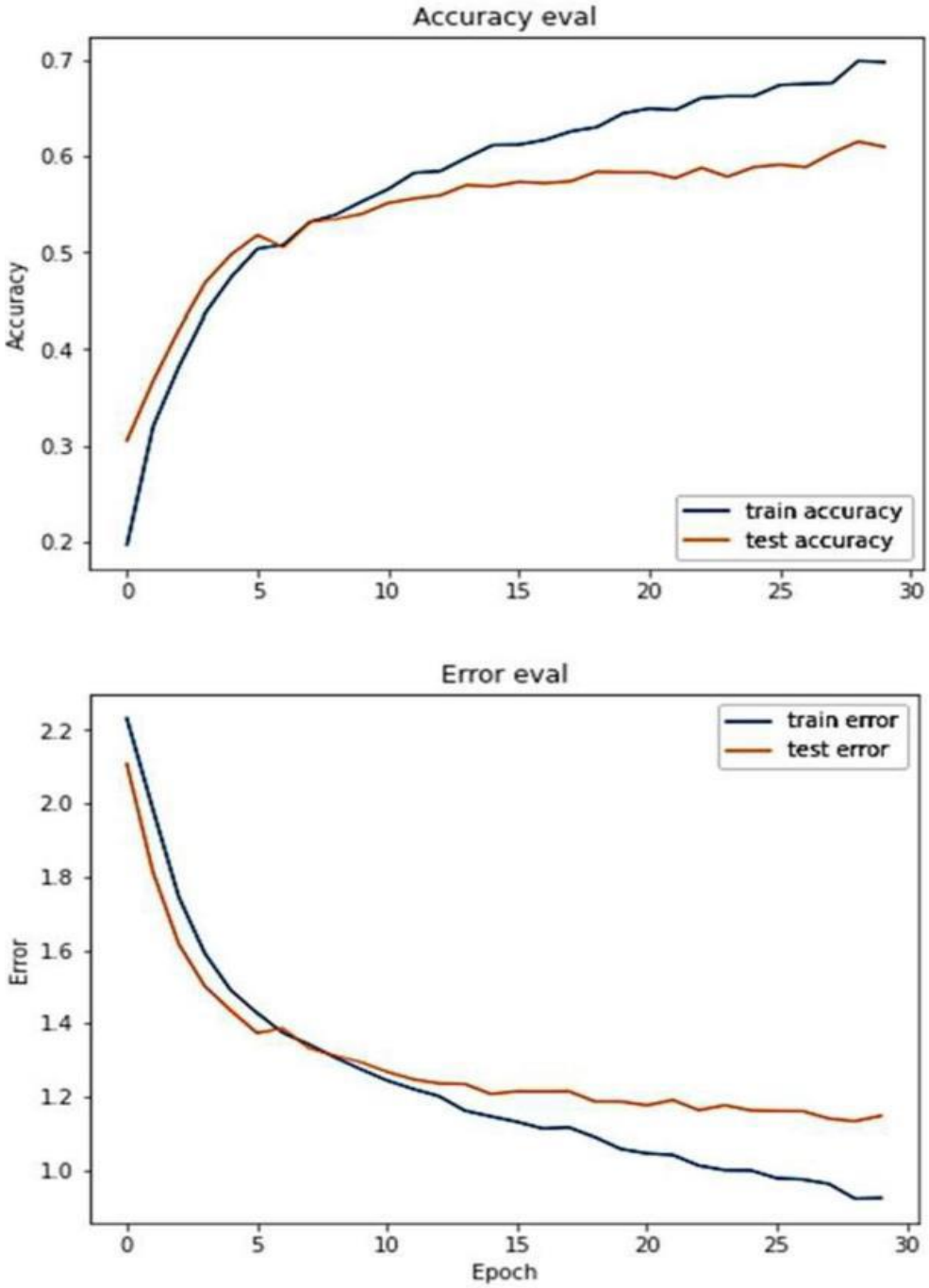
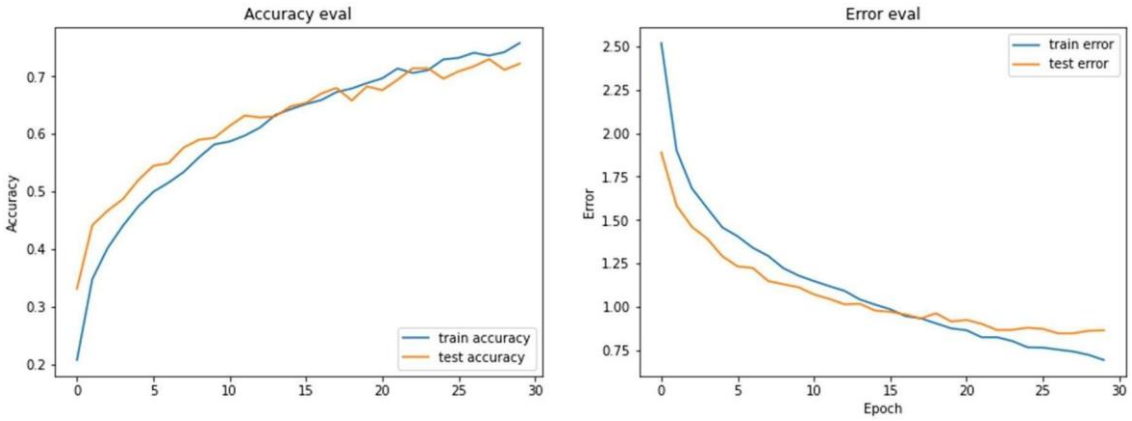


Figure 19: RNN – LSTM Learning cur

Results and Discussion

The model is trained with a dataset in 30 epochs. The accuracy and loss functions are taken into the consideration for evaluation of the model. To do this I have to utilize the model to predict the suitable result on the evaluation dataset and compare that result with predicted target with an actual answer.



Above Figure Compares the both models as achieved on running it on training and testing data set. On training dataset, the model achieved an accuracy of 68.3% while on the test data, it had an average accuracy of 59.07% at the end of 30 epochs. The accuracy is low because of dataset is small.

Conclusion and Future Scope

Music genre classification play a crucial part in music industry as physically rearrangement of knowledge is difficult things. And more even suggestions for comparable music types will likewise be simple. RNN-LSTN is great strategy to use for music kind arrangement because it recalls the previous consequence of the portable within the recurrent layer and characterize music all the more better and productive way.

This study only compared how well a LSTM model would compare to a CNN model when trying to classify the genre of a song, but it might even be interesting to determine how well they might perform at classifying the emotions of a song. having the ability to mention if the song is energetic or happy as an example would give in our own way to assist classify songs when genres start sounding like one another. A follow up study comparing these two models with different features would also help the understanding of which model works best reckoning on the knowledge given.

References

[1] Hareesh Bahuleyan, “Music Genre Classification using Machine Learning Techniques”, Sound (cs.SD); Audio and Speech Processing (eess.AS), 2018

[2] Jeremy Reed and Chin-Hui Lee, “A Study on Music Genre Classification Based on Universal Acoustic Models”, ISMIR, 2006

[3] Thomas Lidy and Andreas Rauber, “Evaluation of Feature Extractors and Psycho-Acoustic Transformations for Music Genre Classification.”, ISMIR, 2005

[4] Kaichun K. Chang, Jyh-Shing Roger Jang and Costas S. Iliopoulos, “On music genre classification via compressive sampling” 2013 IEEE International Conference on Multimedia and Expo (ICME), 2013

[5] Yannis Panagakis, Constantine Kotropoulos and Constantine Kotropoulos “Music Genre Classification Using Locality Preserving Non-Negative Tensor Factorization and Sparse Representations”, ISMIR, 2009

[6] Matan Lachmish, “Music Genre Classification”, https://medium.com/@matanlachmish/music-genre-classification- 470aaac9833d, 2018

[7] Music Genre Classification, https://www.kaggle.com/c/music-genre-classification/datasets

[8] Ms. Sonali. B. Maind and Ms. Priyanka Wankar, “Research Paper on Basic of Artificial Neural Network”, International Journal on Recent and Innovation Trends in Computing and Communication,Vol 2, Issue 1, 2014

[9] Hareesh Bahuleyan, “Music Genre Classification using Machine Learning Techniques”, arXiv.org, 2018

[10] Sumit Saha, A Comprehensive Guide to Convolutional Neural Networks, https://towardsdatascience.com/a-comprehensive-guide-to-convolutional- neural-networks-the-eli5-way-3bd2b1164a53, 2018

[11] Carlos N. Silla Jr., Alessandro L. Koerich and Celso A. A. Kaestner, “A Machine Learning Approach to Automatic Music Genre Classification”, Journal of the Brazilian Computer Society, Vol 14, 7-18, 2008

[12] Apeksha Shewalkar, Deepika Nyavanandi, Simone A. Ludwig, “Performance Evaluation of Deep Neural Networks Applied to Speech Recognition: RNN, LSTM AND GRU”, JAISCR, Vol. 9, No. 4, pp. 235 – 245, 2019

[13] Sepp Hochreiter and Jürgen Schmiduber. “Long Short-Term Memory”, Neural Computation, Volume 9 , Issue 8, 1947

[14] Khamees, A.Ahmad, D. Hani, Alshurideh, M. a. Salloum and S.A, "MUSIC GENRE CLASSIFICATION USING CNN AND RNN-LSTM," proceedings of AMLTA, vol. 1339, no. 1, pp. 315-323, 2021.

[15] Music Feature Extraction in Python https://towardsdatascience.com/extract-features-of-music-75a3f9bc265d

[16] Music Genre Classification using Convolutional Neural Network, https://medium.com/bisa-ai/music-genre-classification-using-convolutional-neural-network-7109508ced47

[17] The Importance of Being Recurrent for Modeling Hierarchical Structure by Ke Tran, Arianna Bisazza & Christof Monz found here: aclweb.org/anthology/D18-1503