International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 08 Issue: 06 | June 2021 www.irjet.net p-ISSN: 2395-0072

Music Information Retrieval and Classification using Deep Learning

Suryansh Jain[1], Shivan Yadav[1], Prakhar Prabir[1], Sundar S[2]

[1]*B.Tech Undergraduate, School of Electronics Engineering, VIT, Vellore, Tamil Nadu, India*
[2]*Associate Professor, School of Electronics Engineering, VIT, Vellore, Tamil Nadu, India*

Abstract - *Music classification is one of the fundamental problems in music information retrieval (MIR). This paper describes the work of building a music genre classifier using deep learning methods. We review several different types of music representation techniques to represent music data in the form that most closely mimics a human auditory system's response to a particular music excerpt; this helps the classifier produce results closely related to how a human brain would classify a music excerpt. Upon extracting and representing the music data in the best possible way in this setting, we then use the extracted information to train and test several different types of deep learning models and compare them on the counts of accuracy in classifying a music excerpt into one of the several different listed genres.*

Key Words: **Music Information Retrieval, Music Genre Classification, Mel-frequency Cepstral Coefficients, Deep Learning, Convolutional Neural Network**

1.INTRODUCTION

Music streaming services have grown profoundly and provided users with a wide variety of songs. Streaming services like Spotify have a catalog of over 50 million songs. This has created a challenge to categorize songs into specific genres. Thus, in this paper, we review methods of classifying music using Deep Learning Neural Networks. We Discuss why Deep Learning is used instead of Traditional Machine learning. We determine the best audio representation method in this setting and then compare several Deep Learning models' performances to classify songs. First, we apply music information retrieval (MIR) techniques to represent song data similarly to the way humans perceive songs. MIR techniques provide a 2D representation of song data similar to the representation of image data, allowing us to use similar methods used in image classification to classify songs. Song data after MIR is directly given as input to the neural network for end-to-end learning. No manual feature engineering is required, which is the advantage of Deep Learning methods over traditional machine learning. The first model we train is Multi-Layer Perceptron (MLP), also known as a vanilla neural network, and can approximate any function. Then we train and compare CNN and RNN-LSTM neural networks. Finally, we determine the best performing deep learning model.

2. RELATED WORK

In the field of Music Information Retrieval (MIR), Deep learning methods have become more prevalent in the last decade. Deep learning for MIR is still a relatively new field compared to the likes of computer vision but has shown much potential. While studying the available literature, we found one of the most informative research papers for a dive into the world of MIR; it describes how deep learning techniques are used in MIR and explains architectures and training of the deep learning networks used [1]. The paper on Machine listening intelligence provides insight into a research framework for music signal modeling, deep learning, and computational musicology [2]. For the dataset, we chose to use the most popular and widely used dataset in the field of MIR - GTZAN. In one of the researches, we found that they formally analyzed the composition of the dataset and also pointed out some of the issues with the integrity of the dataset, e.g., repetitions, mislabeling, distortions, etc. [3]. To extract information from the music excerpts, we used the Python programming language. We also used a python package called *librosa.* Its creators present the paper on Librosa to explain its working and functions that can be used in the field of MIR [4].

Audio Classification using Traditional machine learning techniques involves creating a pipeline of feature extraction and classifier learning. The features are mostly manually designed to succinctly represent acoustic or musical characteristics given the task. We reviewed a music classification model using multi-feature fusion and machine learning algorithms [6]. Humans are particularly good at listening to short samples of songs with the ability to distinguish the artist, the song title, and even the genre. Emulating these abilities has been attempted through a number of NN approaches, and they have shown varying levels of success [18]. Recent breakthroughs in Deep Learning allow us to represent learning in an end-to-end fashion and the task of manual feature engineering can be omitted. The audio data is represented as 2D image-like signals such as Mel-spectrograms, which allows us to use the same techniques of Deep Learning to classify music as used to classify images [5]. The availability of large Datasets which are essential for Deep Learning has risen in recent years, and one such dataset is Million Song Dataset (MSD) which was launched in the year 2011 has motivated the use of deep learning in music genre classifiers. This research article describes the work on using deep learning for music genre classifiers and music recommendation models [20].

Deep Learning models most commonly used for music classification and related tasks are 1D CNN, 2D CNN, Sample Level CNN. Some of the Advanced and most recent neural network architectures used in music classifications are Convolutional Recurrent Neural networks (CRNN), Residual Networks, squeeze and excitation networks.

MLP, also known as a vanilla neural network, is the most straightforward Deep Learning neural network architecture. They do not make any assumptions about the pattern classes in consideration. A two-layer backpropagation network with sufficient hidden nodes has been proven to be a universal approximator [11].

Convolutional neural networks have played an important role in image processing and classifying images by assigning importance or weights to different elements in the image. CNN has been widely accepted due to the efficiency and time to run over other different deep learning models in music genre classification. We have seen research in which the input is taken as a spectrogram of the audio signal, and the model is trained accordingly to differentiate between different spectrograms. We have also seen various models used in combination with different classifiers, such as SVM [7]. Here they have tried to improve the performance of the model by combining classical algorithms with dilated CNN. This method has been widely used in the classification of input images. Another method that is used is extracting different acoustic features by using signal processing and then using CNN for recommendations and genre classification [8]. In one of the research articles, we also found a music genre classification system based on CNN that includes Squeeze & Excitation Block (SE-Block), and then use Bayesian optimization to search the best parameters of SE-Block [15]. We went through numerous researches and also found the use of braided CNN-based neural architecture that learns a sparse representation imitating the receptive neurons in the primary auditory cortex in mammals [16].

1D CNN is one of the earliest advancements in music classification. 1D CNN is efficient and relatively easier to train as the first layer takes the entire frequency range as input, significantly decreasing the number of neurons in subsequent layers and reducing the training parameters; however, this also means this network architecture cannot take advantage of large datasets and hardware advancements.

2D CNNs improve the flexibility of 1D networks. These networks combine smaller time-frequency patterns to create larger ones and perform better in general than 1D CNNs but require better hardware. 2-D CNNs lead to better results in music classification, as they provide more flexibility.

SampleCNN takes raw waveforms directly and has very small sizes of filters. The architecture has proven effective in music classification tasks [12]. It is the first architecture that has achieved state-of-the-art performance with a significantly shorter kernel size than the regular window size in short-time analysis with a deep network.

Another method used for Music classification is the RNN-LSTM. It has been significant in audio classification after extracting the MFCC from the audio files. An audio signal is changed to a vector of different features divided into minor segments on which GRU-based RNN is trained [10]. Another way to use RNN-LSTM is by training it on the independent frame feature, and then the soft-max probability is donated as the LSTM frame representation of music data [9]. By going through various articles and research done in this field, we see that sometimes when the gap is high between the past and the current state, there is a significant loss observed.

3. MUSIC INFORMATION RETRIEVAL

Music information retrieval is one of the fast-growing fields of research. The science and study of extracting information from music are used in many applications, including Music Genre Classification (MGC). In this experiment, we extract information from the music excerpts in the GTZAN dataset and try to find the most suitable representation of the same to be used in our MGR system. A suitable representation of a music excerpt in this setting can be explained as a type of representation that includes information about both frequency and time domain features but also resembles the music excerpt in a way similar to how a human would perceive it.

In the simplest form, an audio signal can be represented as a waveform between amplitude on the y-axis and time on the x-axis. This type of representation of any audio signal has information about the loudness, i.e., the amplitude of the signal at any particular instant of time but does not contain any information about the frequency domain of the audio signal. Therefore, the amplitude vs. time representation of an audio signal can only convey how loud the music or speech signal is at any time but is unable to present any information on how the audio signal would sound and be perceived by a human.

An audio signal can also be decomposed and expressed as a sum of sine waves at different frequencies. This is achieved by performing Fast Fourier transform (FFT) on the audio signal. As a result of FFT, we obtain a power spectrum of the audio signal. A power spectrum represents the distribution of power into frequency components composing that signal, thus providing information about the frequency domain. However, this representation fails to preserve the time-domain information of the signal and hence does not describe how the audio would sound over its duration.

Short-Time Fourier Transform (STFT) is a Fourier related transform that can be explained as a sequence of Fourier transforms of a windowed signal. This allows us to perform a time-frequency analysis of any signal. When performed on an audio signal preserves both the time and frequency domain knowledge of the audio signal simultaneously. The result of STFT can be visualized in the form of a

spectrogram. In this setting, the spectrogram is a two-dimensional graph with a third variable.

Much research in the field of auditory science has also revealed that a human auditory system perceives loudness logarithmically; therefore, we turn our previous spectrogram into a log-spectrogram to produce an audio representation that more closely resembles the human auditory system's response to the audio signal.

Although STFT can preserve both time and frequency domain information of the audio signal, it fails to account for the timbral characteristics of our audio signal. Timber is the feature that enables listeners to distinguish between sounds produced from different musical instruments.

Finally, as a goal of building a music genre classifier, in the next section, we use a form of audio representation that not only carries all the information that we were able to extract from our audio signal using STFT but also accounts for the timbral characteristics of the same.

3.1 Mel-frequency Cepstral Coefficients

Mel-frequency Cepstral Coefficients (MFCCs) are coefficients that collectively make up a Mel-frequency Cepstrum (MFC). The easiest way to understand it is to expand the acronym MFCC. MFCC stands for Mel-frequency cepstral coefficients.

Mel-frequency refers to frequency measured on a Mel-scale. Mel-scale is a scale that relates the frequency of a tone to the actual measured frequency. By measuring frequency on a Mel-scale, we can more closely match the human auditory system's response to change in frequency. Mentioned below in eq (3.1) is the formula used to convert (Hz) hertz into (Mel) mels:

$$mel = 2595 \log10 (1 + Hz / 700) \quad (3.1)$$

Cepstral has derived from the word Cepstrum, and the term Cepstrum was derived by reversing the first four letters of the spectrum. Cepstrum provides us with information about the rate of change in spectral bands. The formula for obtaining Cepstrum can be described as in eq (3.2) -

$$Cp\ (x(t)) = F^{-1}[\log(F[x(t)])] \quad (3.2)$$

In the above equation, x(t) represents the audio signal as a function of amplitude vs time. Cp (x(t)) represents the Cepstrum of the audio signal. The Cepstrum is obtained by a three-step process.

1. F[x(t)] represents the Fourier transform of the audio signal x(t). As a result of the Fourier transform, we obtain a spectrogram of the audio signal.
2. log(F[x(t)]) represents the log-spectrum obtained by converting the scale to a logarithmic scale.
3. $F^{-1}[\log(F[x(t)])]$ represents the Inverse Fourier transform of the obtained log-spectrum. This provides us a cepstrum of the audio signal (x(t)).

Finally, the process of calculating MFCCs from an audio signal can be described as -

1. Convert the audio signal to a time vs amplitude waveform.
2. Perform Discrete Fourier Transform on the waveform to obtain a power spectrum of the audio signal.
3. Convert the spectrum to a log-amplitude spectrum.
4. Perform mel-scaling on the previously obtained spectrum using triangular mel-bands as shown in fig (1). This decorrelates energy into different mel-bands. As seen in the figure, the bands are more concentrated in the lower frequency region and sparser in higher frequencies. This is done to more closely mimic the filter bank in the human ear.

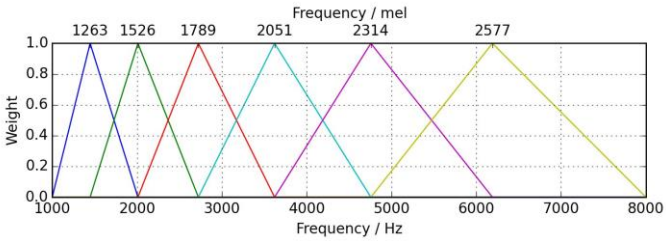

Figure 1: Mel-bands

5. Finally, perform Discrete cosine transform (DCT) on the spectrum that gives us real-valued coefficients called MFCCs. An example of MFCCs visualized can be seen in fig (2).

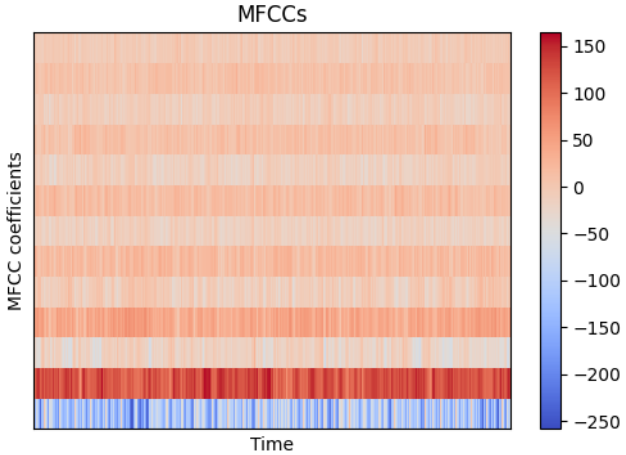

Figure 2: Visualization of MFCC extracted from a music excerpt

4. DEEP LEARNING

Deep Learning is a subset of machine learning in Artificial Intelligence. Deep learning is based on artificial neural networks inspired by biological neural networks found in the brain. Like the human brain, these neural networks use algorithms to identify relationships in a vast amount of data. There are several different ways to implement a deep learning algorithm.

4.1 Activation functions

Activation functions are used to determine the activations of neurons. The activation of a neuron can range from [-1, 1] or [0, 1]. The Activation Functions are broadly divided into two categories Linear and Nonlinear. The activation functions used in our experiments are as described below:

a. Rectified Linear Unit (ReLU)

ReLU is used as an activation function in DNNs, with SoftMax function as their classification function [14]. It is half rectified, i.e., outputs zero for negative inputs, while for positive input, its output increases linearly with input. The function and its first-order derivative are monotonic in nature.

Some advantages of ReLu are -

- Computationally easier to compute than sigmoid
- Solves vanishing gradient problem
- Better convergence functions like sigmoid

Disadvantages of ReLU-

- If too many activations get negative, the networks simply output zero, also known as the dying ReLu problem.
- No upper bound such as sigmoid, the activation can blow up.

b. SoftMax Function

The SoftMax function maps a K dimensional vector to a K dimensional vector such that the components of the output vector add to one. The output vector components can be interpreted as probabilities. The domain of SoftMax is the set of all real numbers while the range is [0, 1]. This is a linear function. The SoftMax formula is described in eq (4.1):

$$\sigma\left(\vec{z}\right)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \qquad (4.1)$$

The above equation, σ represents the softmax function and $\vec{z}$ represents the input vector. The numerator on the RHS of the above equation represents the standard exponential function for the input vector and K represents the number of classes in the classifier. The denominator is the normalization term, ensuring that all the function's output values will sum to 1, thus constituting a valid probability distribution.

An advantage of using SoftMax is it can be used in the output layer of a multiclass classifier since it outputs a probability distribution.

4.2 Optimizers

Optimizers are algorithms used for the faster training of neural networks. Selecting an optimizer is a central step in the contemporary deep learning pipeline. Optimization algorithms are typically defined by their update rule, which is controlled by hyperparameters that determine its behavior, e.g., the learning rate [13]. Some of the most commonly used optimizers are Stochastic Gradient Descent (SGD), Adagrad, AdaDelta, and Adam.

A brief description of each is given below:

a. Stochastic Gradient Descent (SGD)

Hyperparameters are updated for each training example. Network converges faster to local minima and requires more minor memory store losses, and doesn't need to be stored. However, the network may sometimes overshoot after reaching minima. The Hyperparameters fluctuate a lot on their way to achieving stable minima.

b. Adagrad

AdaGrad (Adaptive Gradient) is a modified stochastic gradient descent algorithm. It maintains a per-parameter learning rate. This optimizer decreases the learning rate for dense parameters while increasing the same for sparser parameters. This optimizes training in situations where the output is more dependent on sparse parameters relative to SGD. However, this is computationally more expensive in general as the learning rate always decreases, known as the decaying learning rate problem.

c. AdaDelta

This algorithm is designed to overcome the decaying learning rate problem of Adagrad.

d. Adam

This algorithm calculates Hyperparameters using running averages of both gradients and their second moments. It combines the best properties of the AdaGrad and RMSProp algorithms to handle sparse gradients on noisy problems.

4.3 Deep learning models

a. A Multilayer perceptron (MLP)

MLPs are feedforward artificial neural networks. Neural networks are a set of algorithms modeled after the human brain. They are used for pattern recognition on numerical input data and output a label [17]. MLPs have been proven to be a universal approximator. It is trained using the supervised learning method known as backpropagation. It is a fully connected network, i.e., every neuron in the previous layer is connected to every other neuron in the next layer with at least three hidden layers. In general, it has an input layer followed by several hidden layers, and finally, the output layer.

b. RNN-LSTM

A recurrent neural network works on the basic formula of the new state of RNN at time t is a function of its old state and the input at time t. In RNN, the same set of weights are recursively applied on a model. The output of the previous state is the input for the next state, as shown in fig (3). If the weights are significantly less, we get the maximum loss; this is known as vanishing gradient. To reduce this loss, we need to apply LSTM. LSTM has three gates, forget, input and output. Each gate has different sets of weights. The output of the gates is determined through a set of networks inside the cell with n number of units. The old states and inputs are multiplied by the respective weights of the given gates. The output is then passed through an activation function.

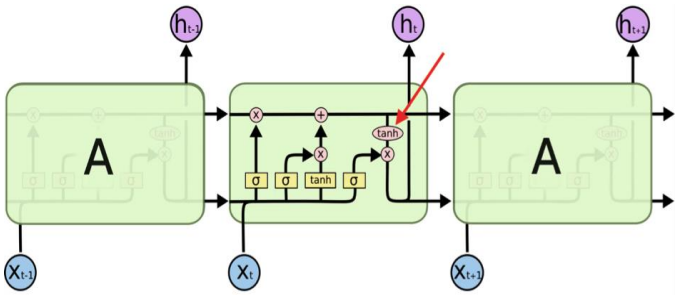

Figure 3: RNN-LSTM architecture

c. CNN

A convolution neural network is a deep learning algorithm mainly used to extract different components of an image and classify it. The algorithm assigns different weights or biases to the various aspects of an image, making it different from another image. CNN consists of one or more hidden layers referred to as convolutional layers. The basic aim of these hidden layers is to detect a pattern in an image using various filters (kernels) which perform convolution on the input image. An input image goes through various hidden layers and various filters, after which the output of the classification is received, as shown in fig (4). Filters consist of the neurons in a convolutional layer that covers the entire input and looks for one feature. These neurons are connected by weights.

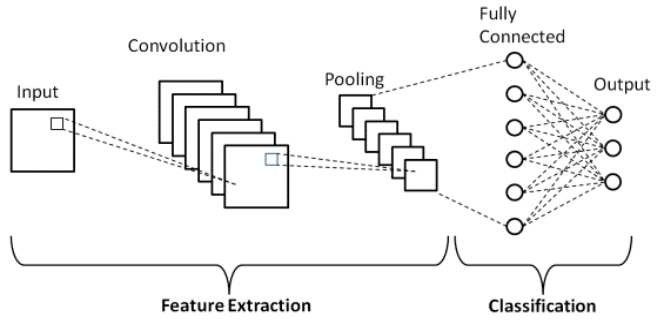

Figure 4: CNN architecture

5. DATASET

The dataset of choice for our experiment is the GTZAN dataset. G Tzanetakis assembled the GTZAN dataset in 2001, and it has been widely used in the field of Music genre recognition (MGR) since. The GTZAN dataset consists of 1000 music excerpts of 30 seconds duration with 100 examples in each of 10 different music genres: Blues, Classical, Country, Disco, Hip Hop, Jazz, Metal, Popular, Reggae, and Rock. The composition, integrity, and issues like mislabeling, replications of the dataset have been formally analyzed.

6. LIBROSA

Librosa is a Python package for audio and music signal processing. Librosa provides implementations of a variety of standard functions used throughout the field of music information retrieval. As seen in the previous section, calculating MFCCs from music excerpts can prove to be a herculean task and is prone to error. In this experiment, we use librosa to extract MFCCs from our GTZAN dataset. The *librosa.feature* module offers various implementations of spectral representations. One of the features in the package is MFCC which can be accessed using the function call (*librosa.feature.MFCC*). The MFCC feature takes various inputs, including the audio signal, sampling rate, etc. These and other input attributes are explained in the next section.

7. EXPERIMENT

7.1 Experimental Overview

The goal of this experiment is to extract information from the GTZAN dataset and then use the retrieved information to train several types of deep learning models to classify any audio file into one of the ten genres in the GTZAN dataset to then study and compare the results from different models on the counts of accuracy and loss function. Figure (5) describes the workflow for our experiment.

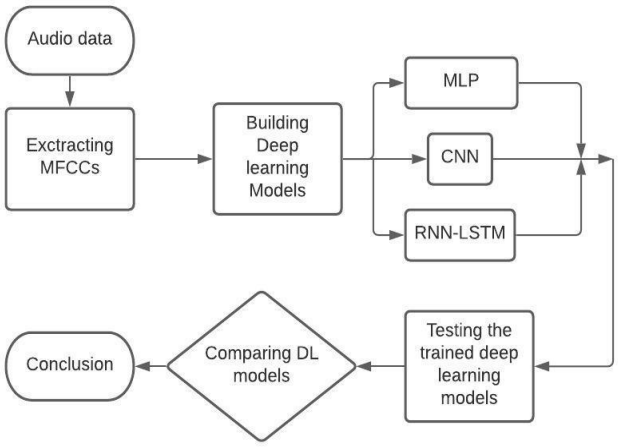

Figure 5: Flowchart describing the workflow for our experiment.

7.2 MIR

We start by processing the GTZAN database and extracting MFCCs for each of the music excerpts present in the dataset. We divide each music excerpt into ten segments of equal time duration. To process the dataset, we use the

python.os package. Next, we use the python.librosa package to extract MFCCs for each segment of every music excerpt in the dataset.

The MFCCs are calculated at the following configuration:

Sampling rate = 22050, Number of fft = 2048

Hop length = 512 , Number of mfc coefficients = 13.

This gives us 13 coefficient values each for 10000 segments (10 segments each for 1000 music excerpts) in the dataset, totaling 130000 data points. Once we have obtained all the MFCCs, we then use the *python.json* package to dump this data into a JSON file which is to be used in the next phase of our experiment to train the deep learning models.

7.3 MLP

We compiled an MLP in Keras with three hidden layers where the first hidden layer had 512 neurons, and the second layer had 256 neurons, while the last hidden layer had 128 neurons. The Output layer has neurons equal to the total number of genres a song can be classified into, which is 10. ReLu is used as the activation function for input and hidden layers, and neurons in the last layer use the SoftMax activation function. For our experiment, the dataset was divided into two parts, 30% was used for training of the model and the remaining 70% for testing of the trained model. The above network performed very well on training data with around 90% accuracy after 50 epochs, but the performance on test data was erratic. The accuracy oscillated around the same value indicating overfitting on test data. Overfitting can be solved using various techniques such as data augmentation, removing hidden layers, or randomly dropping neurons in each hidden layer for each epoch so that each neuron generalizes well over the training data. We took the last approach in our experiment. Modifying the training of the network in such a way resulted in test accuracy steadily increasing along with training accuracy after each epoch instead of oscillating.

7.4RNN-LSTM

We have used two RNN-LSTM layers with 64 units. Then we have added a dense layer with ReLU activation and a dropout layer with a rate of 30%. The output of the dense layer is then put into the output layer with the SoftMax activation function. Finally, we have compiled the model with an optimizer function with a 0.0001 learning rate. The final test accuracy that we get is around 64%.

7.5 CNN

We start with building a sequential model. A sequential model allows creating the model one layer after another. In this model, we have used three convolutional or hidden layers with different arguments. An activation function decides if a neuron should be activated by calculating the weighted sum. The set of filters, their dimensions, and the activation function of different layers are mentioned in table (1).

Table 1: Properties of CNN

Convolution Layer	Activation function	Filters	Filter dimensions
Layer 1	ReLU	32	3x3
Layer 2	ReLU	32	3x3
Layer 3	ReLU	32	2x2

In between the convolutional layers, we add the max-pooling function along with padding all the edges that down sample the input.

Pooling function: 3x3 grid size, 3x3 strides

After pooling, the batch normalization function is used to normalize the activations in the current and subsequent layers. The main purpose of using this function is to speed up the training of the models. Next, the output is flattened to fit into a 1D array, and we add a dense layer with 64 neurons. Now to avoid overfitting, we use the dropout function with a probability of 30%, the outputs of the layers under dropout are randomly subsampled. Thus, it has the capacity of thinning the network during training.

The final output layer is a dense layer.

Number of neurons: 10

Activation function: SoftMax.

It creates scores for each neuron or category. It takes the index with the highest value and maps it to the relative genres. Finally, all the inputs are classified into one of the ten genres.

8. RESULTS AND DISCUSSION

In this experiment, we compare the performance of three deep learning models on the counts of accuracy and test error over 100 epochs. The accuracy is measured on the scale of [0,1], where 1 depicts a 100% accurate system. The error is measured by calculating the Root Mean Square Error, and the scale ranges from 0 to infinity, where 0 represents no error in the output.

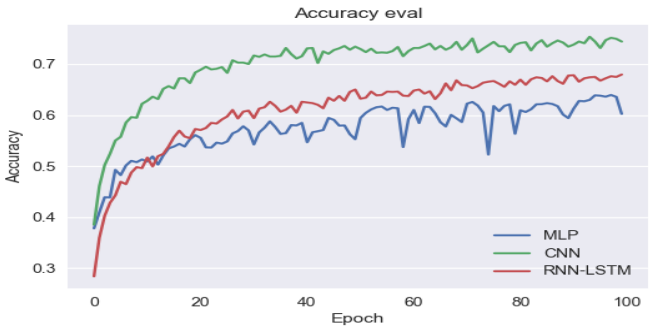

Figure 6: The accuracy trend over 100 epochs for each of our three deep learning models.

Figure (6) shows the accuracy trend for our deep learning models over 100 epochs. We observe that the overall trend for the accuracy value for MLP increases for the test dataset; however, it varies and fluctuates significantly, indicating it struggles to find a local minimum. For CNN, we see the curve saturating around 80 epochs. For RNN-LSTM, we observe the same trend as CNN but at a lower accuracy value.

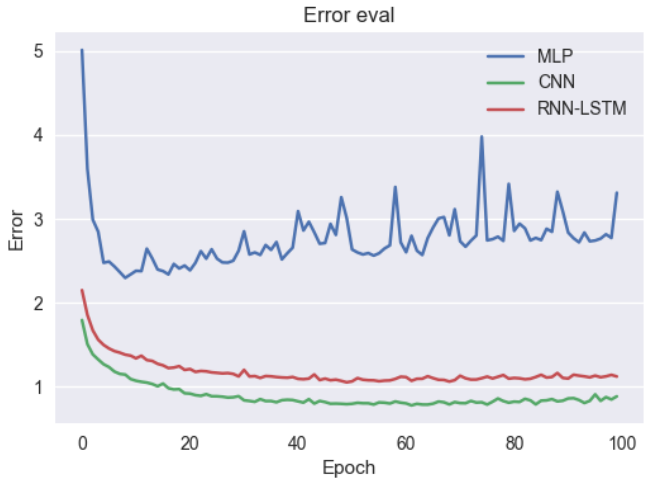

Figure 7: The Error trend over 100 epochs for each of our three deep learning models.

Figure (7) shows the error trend for our deep learning models over 100 epochs. For the error evaluation, we see a similar trend as in the accuracy evaluation. MLP produces a fluctuating error plot with the error value in the range of 2.5 to 4. However, for the other two models, CNN and RNN-LSTM, we observe much smoother curves with lower error values.

On comparing CNN, MLP, and RNN-LSTM models, we observe CNN outperforming MLP with a margin of almost 15%. This can be explained by the fact that the MLP is inefficient due to redundancy in high dimensions. In MLP, the total number of parameters can be very high. In the RNN-LSTM model, we see a significant drop in loss compared to CNN due to high gaps in the previous and present state. In CNN, every node does not connect to every other node. They are sparsely connected rather than fully connected. Also, in CNN, the weights are small, and some weights are even equal to zero, and at every step, the process of sub-sampling takes place, making the layers smaller and smaller. This eliminates the redundancy factor, eliminates the loss in CNN, and allows CNN to be much more efficient and faster than MLP and RNN-LSTM. Although when we compare the RNN-LSTM model to the MLP model, we get better accuracy in RNN-LSTM because it stores the memory of the previous data and uses it as an input for the next successive step.

The test accuracy and test error of different models are mentioned in table (2).

Table 2: Accuracy and Error evaluation

Model	Test Accuracy	Test Error
MLP	0.6024	3.31
RNN-LSTM	0.6795	1.12
CNN	0.7443	0.88

9. CONCLUSION

In this paper, we compared several time-frequency audio representation techniques and listed their advantages over one another, concluding to use MFCCs to extract information from our GTZAN dataset and feed the data to the deep learning models. Among all the neural networks tested in our experiment, CNN performs the best with around 75% accuracy. In comparison, the RNN-LSTM network performs significantly better at an accuracy of 67.95% than the MLP network, which has an accuracy of 60.24%.

When trained on a large dataset such as Million Song Dataset (MSD), such neural networks can produce more accurate results and require more computational capacity. When trained with a larger dataset, these networks can also be used for other applications like making music recommendations or predicting hit songs using transfer learning techniques.

10. FUTURE WORK

MGC algorithms are commercially used in platforms like Spotify, YouTube music, etc. An improvement in our project can be made using the user behavior parameter while listening to any music excerpt (playtime, skips, repeats, etc.) and use these to alter the weight of the respective nodes in our model to finally create a playlist that contains similar songs (genre) as given in the input but also takes into account user's song listening history as discussed above, thus generating personalized playlist for each user. Another application is music thumbnailing which involves finding the most interesting part of a song. This can be achieved by training another neural network on top of a classifier using transfer learning. Transfer learning consists of a source task and a target task. The neural network trained in the source task can be reused in the target task after adapting the network to a more specific dataset. This has been gaining more attention in music informatics research for alleviating the data sparsity problem [19].

REFERENCES

1. Choi, K., Fazekas, G., Cho, K., & Sandler, M. (2017). A tutorial on deep learning for music information retrieval. arXiv preprint arXiv:1709.04396.
2. Cella, C. E. (2017). Machine listening intelligence. arXiv preprint arXiv:1706.09557.
3. Sturm, B. L. (2012, November). An analysis of the GTZAN music genre dataset. In Proceedings of the second international ACM workshop on Music information retrieval with user-centered and multimodal strategies (pp. 7-12).
4. McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., & Nieto, O. (2015, July). librosa: Audio and music signal analysis in python. In Proceedings of the 14th python in science conference (Vol. 8, pp. 18-25).
5. Sun, B. (2021). Using Machine Learning Algorithm to Describe the Connection between the Types and Characteristics of Music Signal. Complexity, 2021.
6. Pelchat, N., & Gelowitz, C. M. (2020). Neural Network Music Genre Classification. Canadian Journal of Electrical and Computer Engineering, 43(3), 170-173.
7. Nam, J., Choi, K., Lee, J., Chou, S. Y., & Yang, Y. H. (2018). Deep learning for audio-based music classification and tagging: Teaching computers to distinguish rock from bach. IEEE signal processing magazine, 36(1), 41-51.
8. Elbir, A., & Aydin, N. (2020). Music genre classification and music recommendation by using deep learning. Electronics Letters, 56(12), 627-629.
9. Lang, B. (2005, September). Monotonic multi-layer perceptron networks as universal approximators. In International conference on artificial neural networks (pp. 31-37). Springer, Berlin, Heidelberg.
10. Li, H., Xue, S., & Zhang, J. Combining CNN and Classical Algorithms for Music Genre Classification.
11. Costa, Y. M., Oliveira, L. S., & Silla Jr, C. N. (2017). An evaluation of convolutional neural networks for music classification using spectrograms. Applied soft computing, 52, 28-38.
12. Xu, Y., & Zhou, W. (2020, December). A deep music genres classification model based on CNN with Squeeze & Excitation Block. In 2020 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC) (pp. 332-338). IEEE.
13. Sinha, H., Awasthi, V., & Ajmera, P. K. (2020). Audio classification using braided convolutional neural networks. IET Signal Processing, 14(7), 448-454.
14. Kim, T., Lee, J., & Nam, J. (2019). Comparison and analysis of sample CNN architectures for audio classification. IEEE Journal of Selected Topics in Signal Processing, 13(2), 285-297.
15. Phan, H., Koch, P., Katzberg, F., Maass, M., Mazur, R., & Mertins, A. (2017). Audio scene classification with deep recurrent neural networks. arXiv preprint arXiv:1703.04770.
16. Dai, J., Liang, S., Xue, W., Ni, C., & Liu, W. (2016, October). Long short-term memory recurrent neural network-based segment features for music genre classification. In 2016 10th International Symposium on Chinese Spoken Language Processing (ISCSLP) (pp. 1-5). IEEE.
17. Agarap, A. F. (2018). Deep learning using rectified linear units (ReLU). arXiv preprint arXiv:1803.08375.
18. Choi, D., Shallue, C. J., Nado, Z., Lee, J., Maddison, C. J., & Dahl, G. E. (2019). On empirical comparisons of optimisers for deep learning. arXiv preprint arXiv:1910.05446.
19. Lau, D., & Ajoodha, R. (2020). Music Genre Classification: A Comparative Study Between Deep-Learning And Traditional Machine Learning Approaches.
20. Liang, B., & Gu, M. (2020, August). Music genre classification using transfer learning. In 2020 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR) (pp. 392-393). IEEE.