Music Genre Classification using Neural Network and Data Augmentation

Arka Ghosh
Faculty of Computer Science
Dalhousie University
Halifax, Nova Scotia, Canada
arka.ghosh@dal.ca

Shiyun Wang
Faculty of Computer Science
Dalhousie University
Halifax, Nova Scotia, Canada
sh776410@dal.ca

Md. Monzurul Amin Ifath
Faculty of Computer Science
Dalhousie University
Halifax, Nova Scotia, Canada
monzurul.amin@dal.ca

Emmanuel Etukudo Etti
Faculty of Computer Science
Dalhousie University
Halifax, Nova Scotia, Canada
em244217@dal.ca

***Abstract*—Music has implanted the seeds of feelings and emotions ever since the melodious sound of the instruments has fell on the human ears. While music is ultimately compilation of melodious sound that links all hearts around globe, there are further classifications of music genre that bring music lovers together and keep the melody rolling. Although human can categorize the music into specific genre they usually listen to due to the human auditory system, the same cannot be said in case of machine we use. Growing interest in the music genre classification for the orderly grouping of music based on genre has drawn the attention of many researchers recently. This work proposes two Neural Network for music genre classification among which the first one being the Feed Forward Network and the second one being the Long Short Term Memory (LSTM). Mel-frequency Cepstral Coefficients (MFCC) and Chroma STFT is used o extract features from raw audio data which is fed into both the classifiers as an input. To tackle the data scarcity problem while training neural networks, two data augmentation techniques are also proposed. Evaluation on the GTZAN dataset indicates that both the models are capable of classifying music genres, where the Feed Forwarded Neural Network yields higher accuracy of 69.77% when it is trained on the augmented data with MFCC features than the LSTM model that provides an accuracy of 65% on the augmented data with the Chroma STFT.**

***Index Terms*—Music Genre Classification, Music Information Retrieval (MIR), GTZAN dataset, Convolutional Neural Network, Mel-frequency cepstral coefficients (MFCC)**

I. Introduction

With the rapid advancement in the field of technology and internet, people have started to upload music originally stored in vinyl records or CDs to streaming platform on internet. The widespread use of the internet, over the years, has paved the way for significant advancements and changes in the music industry. One notable example of these developments is the widespread use of online music streaming platforms. Although the popularity of the music streaming platforms is in rise in recent years, the vast online music library makes it difficult for people who search for specific genre or music. Additionally, music genres have the potential to significantly develop comprehension. Recognition and appreciation of the music and the artists when used flexibly and descriptively, rather than as a means of rigid division [1]. For this reason, a tool or framework which can classify the songs based on its genre and recommend music to the users based on their genre preference using techniques, i.e. meta-data analysis or collaborative filtering [2], is an important issue.

Along with the recent improvements in the Music Information Retrieval and the rapid increase of music streaming platforms, there has been a surge in demand for music genre classification. The vast amount of music on the online music platform also makes it infeasible and difficult to perform a manual organization of the music based on its genre [3]. As a result, automatic classification and categorization of the music based on its genre is crucial in the Music Information Retrieval as it has practical application, such as organizing the music data effectively based on some specific information or tags. For music streaming platforms, such kind of classification of music genres aid users in expanding their music libraries and providing appropriate suggestion to their users. However, efficient and reliable automatic music information processing remains the primary concern, and it has drawn the attention of an increasing number of scholars, musicians, and composers. The difficulty of organising, characterising, and categorising music materials on the internet is a current on-going topic in automated music information retrieval.

Machine learning algorithms have been applied to handle the Music Genre Classification since 2002, with the initial classification accuracy of 60% being the standard [2]. However, with the significant improvement in the deep learning frameworks, it has been applied in categorizing music based on genres as it can handle huge database and yield more accuracy on the classification tasks. Music Genre Classification is composed of two steps which are feature extraction and building a classifier [4]. In the first stage, various spectral features are retrieved from the raw audio files. As music belonging to the same genre are formed of using similar kinds of instruments, have comparable rhythmic patterns and have similar pitch distribution, audio signal of such groupings shares some certain common characteristics [5]. As a result, the extracted features must be comprehensive accurately reflecting the music, compact and effective for automatic music genre classification. The second stage is focused on building a classifier that utilizes the extracted features from a set of training data to understand and produce an output based on how given input features relate to particular class or group.

In this paper, we have proposed an approach for music genre

classification using simple Feed Forward Neural Network and Long Short Term Memory. While other research is solely dependent on music genre classification, we have focused on both music genre classification, and data scarcity problem in building deep learning classifiers for which we have used data augmentation to generate new data from the existing dataset.

The rest of the paper is organized as follows. Some previous work on music genre classification is presented in the Section 2. Section 3 deals with the methodology used to classify music genre, similarly section 4 is focused on the dataset used, the pre-processing of the dataset and experimental results,, and section 5 describes the conclusion and possible future outcome of the proposed methodology for music genre classification.

II. Related Work

As the music industry expands fast due to new technical advancements, the interest of the researchers in this field of classification grows. Different researchers have suggested various deep learning algorithms to categorise various genre of music. In 2002, on the GTZAN data set, Tzanetakis et al. [2] used time-frequency (TF) analysis approaches such as Mell Frequency Cepstral Coefficient (MFCC), Spectral Centroid, and wavelet modifications in the feature extraction process. The authors have been able to achieve an accuracy of 61% on GTZAN dataset using these acoustic features. Tzanetakis et al. [7] proposed another work on the same dataset by employing Daubechies wavelet coefficient histogram as the learning feature which has achieved an overall accuracy score of 79.5%.

Silla et al. [8] proposed an approach using multiple feature vectors which was chosen from several time segments from the beginning, middle, and end of the music, using a pattern recognition ensemble technique and a space-time decomposition dimension. Naive-Bayes, decision trees, k Nearest-Neighbors, SVMs, and Multilayer Perceptron Neural Networks were used in this approach. When utilising Round-Robin on Space-time ensemble, the best accuracy attained was 65.06%. Panagakis et al. [9] proposed a framework for music genre classification which considers the crucial properties of the auditory human perception system, namely 2D auditory temporal modulations representing music and sparse representation. The accuracies the authors have got is higher than any accuracy previously reported for the GTZAN and ISMIR2004 datasets which are 91% and 93.56% respectively.

During the initial stages of the introduction of Convolutional Neural Network into the music genre classification, a verification has been made by Sigtia et al. [10] where the authors have showed that ReLU, Dropout and Hessian-Free optimization can result into a better feature learning effect. To fully use the characteristics of CNN's feature extraction, LH et al. [11] trained CNN as a feature extractor first, and then integrated the majority voting method to train feature-based classifiers, yielding considerable results on the GTZAN dataset. Xu [12] et al. proposed a novel approach for music genre classification based on Convolutional Neural Network to completely mine the latent information contained in the input spectrum graph. This approach incorporates the Squeeze and Excitation Block (SE-Block) followed by the Bayesian optimization to find the optimal SE-Block settings. To evaluate the model, the authors utilized the GTZAN dataset and achieved a overall classification accuracy of 92%. Bisharad et al. [13] proposed a Residual Neural Network which is trained on the 3 seconds audio samples collected from the GTZAN dataset. Melspectrogram is computed for each audio files using a sliding Hanning window of 20 ms with 50% overlap and 128 Mel co-efficient which produces a feature vector of 300x128 for each genre. This feature vector is fed into the RNN which has been able to achieve an accuracy of 82%, 91%, and 94.5% for the top-1, top-2, and top-3 most probable genre classes assigned to an audio sample at test time, respectively.

Given these, it is visible that a lot of work has previously been done in Music Genre Classification using traditional Machine Learning and Deep Learning frameworks on the GTZAN dataset. However, most of the approaches are heavily focused on the rhythmic or pitch content, feature extraction and the architecture of the model. As illustrated before, our work explores the different augmentation techniques on the GTZAN dataset which is derived from the original data with some minor transformation to increase more examples and variety in training dataset so that classifier can learn the feature from more data points.

III. Methodology

A. Dataset Used

In order to build and evaluate our model, the GTZAN [2] dataset has been used. This dataset has been used widely as a benchmark dataset [6] in machine learning and deep learning research for music genre recognition. GTZAN is a compilation of 10 genres with 100 audio files per genre illustrated in Fig. 1, each audio having a length of 30 seconds. The tracks are all 22050Hz Mono 16-bit audio files in .wav format. The dataset is divided into 10 genres i.e., classical, jazz, pop, hip-hop, metal, disco, country, blues, rock, reggae. In order to reflect a diversity of recording settings, the files were acquired in 2000-2001 from a number of sources, including personal CDs, radio, and microphone recordings [14]. The audio files are fairly balanced across each genre and the data set has been used in many music genre classification researches previously which is one of the reasons behind choosing the data set for this project.

B. Data Pre-processing

To begin with the audio data processing, it is necessary to adapt a way to concisely visualize the audio files over the time domain representation. For the visual representation of the audio files, waveform are an effective way to plot the audio files as time on the x-axis and amplitude on the

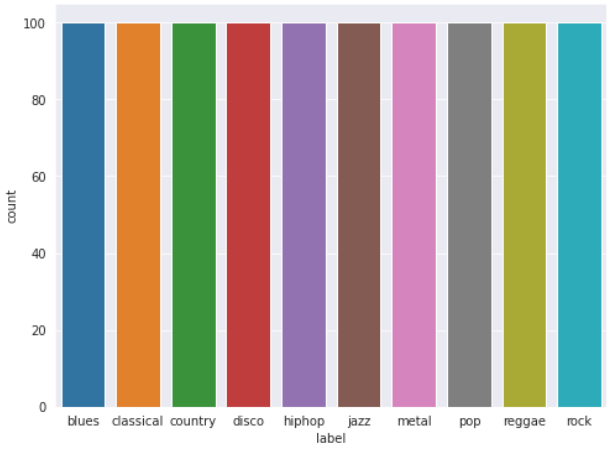

Fig. 1. Distribution of audio files per genre

y-axis. They are excellent for swiftly scanning audio data and graphically comparing which genres may be more similar than others [15]. Some genres of music can be clearly recognised from others when represented in time domain, as seen in Fig. 2. In this regard, Jazz differs significantly from Disco and Pop music. However, several forms of music might have extremely similar characteristics which is hard to recognize from time domain representation, making music genre classification a difficult endeavour, as seen in Fig. 2. In comparison to Jazz, Disco and Pop are significantly more similar, making classification more challenging. Additionally, This approach frequently oversimplifies audio data, which consists of more than just amplitude over time [16]. This is why a more appropriate way is to transform these waveform into spectrogram.

A spectrogram depicts the strength of a signal over time at different frequencies of a waveform. In any kind of music analysis, a signal's frequency domain plays a crucial part as it carries the detailed information through calculating the signal's frequency spectrum [17]. Visualizing audio data with a spectrogram reveals hidden insights about the data that may have been missed in typical waveform representations, helping differentiate noise from actual audio data [16]. A spectrogram's color is either brighter or darker, and it is measured in decibels. The spectral features of visualized audio signals can be used by deep learning frameworks to evaluate and interpret the results. To transform the raw audio into spectral representation, the continuous waves of the sounds are converted into 1D numpy array of digital value by sampling them at discrete interim [18], as illustrated in Figure 3 [19].

These waves are one-dimensional and can be used to express a specified frequency or amplitude at specific time intervals. As a result, a precise sampling rate is employed, and these sample values can be collected if the audio must be rebuilt. However, *LibROSA*, a python package for audio processing, is used for this project which uses a default sample rate of 22050 Hz that helps to keep the array size minimal by lowering

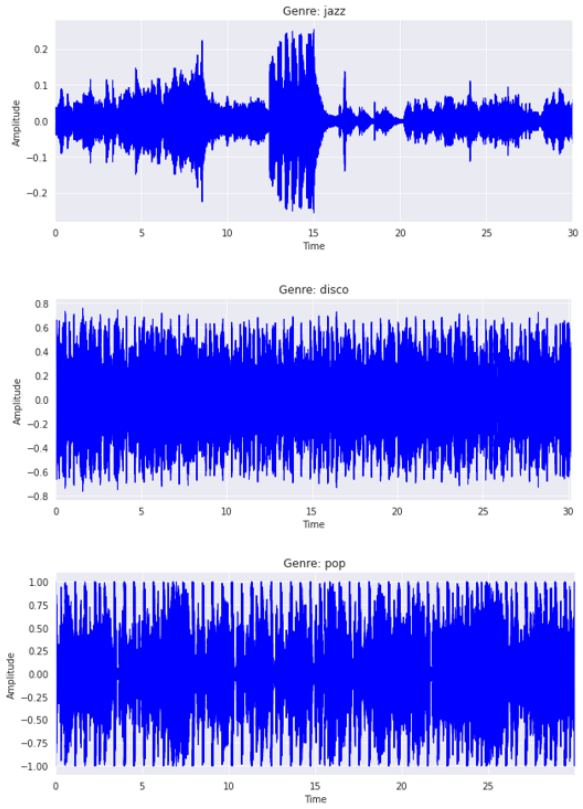

Fig. 2. Time Domain Representation of Jazz, Disco and Pop

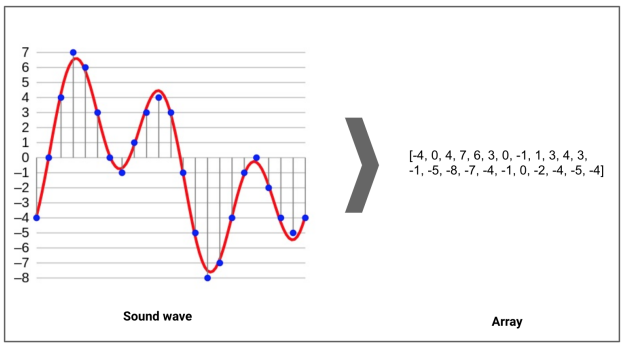

Fig. 3. Digital Representation of a Sound Wave, in red, with the resulting array on the right

the audio quality but increasing the training time. It merges the two stereo audio channels into a one-dimensional numpy array, resulting in a mono audio stream when the audio files are loaded using the *librosa.load()* function. Finally, spectral features are extracted from the audio files once it has been loaded and represented in an array of values.

C. Extracted Spectrogram

To extract the spectral features from the audio files, two spectrogram has been used namely Mel Frequency Cepstral Co-efficient (MFCC) and Chroma STFT which takes the audio file and convert it from time domain to frequency domain using Fourier Transformation. The reason behind choosing

this two spectrogram is that these spectral features are able to produce dominant classification result and effective in error reduction and producing a robust feature even when the signal is influenced by noise [20]. The process of extracting features from an audio data is greatly simplified by the LibROSA module, which has certain built-in Python methods for generating the requisite spectrogram which is described in the following:

1) Mel Frequency Cepstral Coefficients (MFCC): : In order to generate MFCC spectrogram through LiBROSA, the loaded audio, the sample rate of the audio which is 22050Hz by default in our case, and the number of MFCC coefficient need to be passed to the librosa.feature.mfcc() [21] function. Signal features in the time frequency domain of Disco and Pop music appear to be more distinct, as demonstrated in Figure 4. Even though higher order coefficients imply greater degrees of spectral complexity, the models get increasingly complicated when a high number of cepstral coefficients are used for which we have used co-efficient value of MFCC, represented by n_mfcc parameter to 20 [23]. The mean of value of the co-effienct is taken into consideration that converts the two-dimensional array to a one-dimensional array. As a consequence, by taking into account both time and frequency domain representations, more useful characteristics can be extracted from audio data which is helpful in classification of the genres.

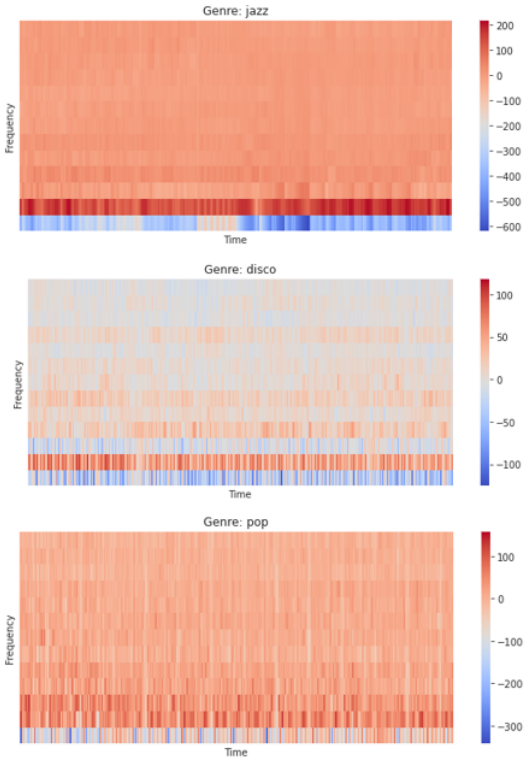

Fig. 4. Time-Frequency Representation (MFCC) of Jazz, Disco and Pop

2) Chroma STFT: The chroma representation describes each of the 12 unique musical chroma of the octave's intensity levels at each time interval. STFT is used to compute the chroma characteristics, illustrated in figure 5. According to the graph's colour bar net, it indicates the spike with high values in low values (dark regions). In order to generate STFT features through LiBROSA, some arguments need to be passed to the feature.chroma_stft() [22] method, including the loaded audio, the sample rate of the audio which is 22050Hz by default in our case, and the hop length to return. Through using Chroma STFT, we have generated the following time-frequency representation of Jazz, Disco and Pop music genre in the following:

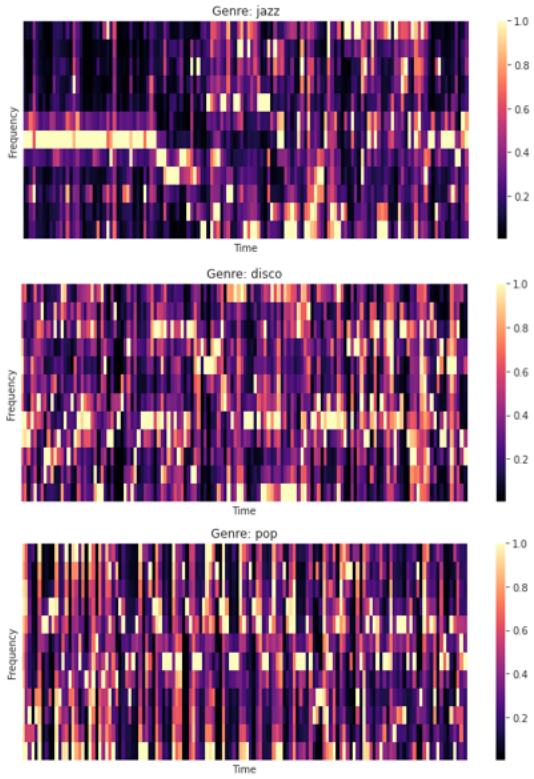

Fig. 5. Time-Frequency Representation (Chroma STFT) of Jazz, Disco and Pop

D. Data Augmentation

Deep Learning algorithms have become the most popular method for image analysis and classification because of the ability of these models in classifying image data. In several classification tasks, Deep Learning models i.e., Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) produced state-of-the-art results, yet despite its broad viewpoints, they still face significant difficulties. Such kind of models are primarily driven by the enormous scale of the networks, which may have millions of parameters, as well as by the dearth of trustworthy training data sets [24]. To prevent overfitting, these networks mainly rely on massive data. The

problem known as overfitting occurs when a network learns a function with a relatively large variance to precisely describe the training data [26]. To tackle the problem of overfitting, data augmentation is a novel approach. The concept of data augmentation is very straightforward; it is the technique of creating additional data points from current data to artificially increase the amount of data. To amplify the dataset, this includes making small adjustments to the data or utilising machine learning models to produce new data points in the latent space of the original data. This process helps in combating overfitting as the model is unable to overfit all the samples when more data points are added, forcing it to generalise [27]. For this work, we have used two effective methods for data augmentation in our project which are adding Gaussian Noise and time shifting over the original GTZAN dataset.

1) Gaussian Noise Injection: We have added white Gaussian noise with a mean value of 0 and a standard deviation value of 1. The mean identifies the curve's peak about x-axis and the variance identifies how broad or spread-out the values are from this point [28]. As the power spectrum of white noise is evenly distributed across all permitted frequencies [29] throughout an audio file, it aids in creating data points from the original data. The wave plot of three different genres after adding Gaussian White Noise is illustrated in Figure, where the orange spikes denote the original data's audio signal and blue spikes denotes the newly generated data with injected Gaussian White. As illustrated in Figure 6, it is visible that white noise can enhance the amplitude of audio waves.

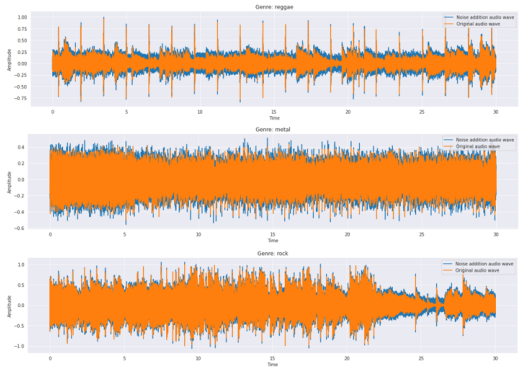

Fig. 6. Time Domain Representation of Reggae, Rock and Metal after Gaussian Noise Injection

2) Time Shifting: The concept of time shifting audio data is simple; slightly shifting the starting point of the audio file and padding it to the original length of the data afterwards. For shifting the time of the audio files, we have shifted the audio save by a sample_rate/2 factor that shifts the wave along the time axis to the right by a specified amount. The wave plot of three different genres after time shifting is illustrated in Figure 7, where the orange spikes denote the original data's audio signal and blue spikes denotes the newly generated data after time shifting. Unlike the Gaussian noise in demonstrated in Figure 6, Figure 7 does not change the amplitude of the audio waves, it just shifts the wave of the audio to the right.

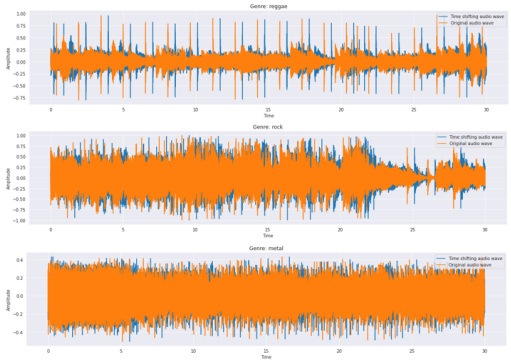

Fig. 7. Time Domain Representation of Reggae, Rock and Metal after Time Shifting

E. Model Architecture

In this section, we describe different model architectures that we try. It includes two parts: neural network and recurrent neural network. For both subsections, we illustrate different layers, activation functions, and the architecture pipeline.

1) Neural Network: The architecture of the first neural network is a simple feed forward network with 4 layers. The first layer is a dense (fully-connected) layer with 256 units and a rectified linear unit (ReLU) activation function. Input data shape varies based on the training dataset used in different models. The second and third layers are also dense layers with 128 and 64 units, respectively, and ReLU activation functions. These layers further process the output from the first layer to extract higher-level features. The fourth and final layer is a dense output layer with 10 units and a softmax activation function. This layer produces the final output of the network, which is a probability distribution over the 10 possible classes.

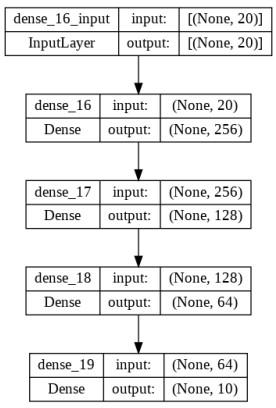

Fig. 8. NN Model Architecture with Adam optimizer

The second neural network architecture is a feedforward network with 6 hidden layers and one output layer. The first layer is a dense (fully-connected) layer with 1024 units and a ReLU activation function. This layer takes in the input data with a shape of (,20), where the input is the training data for the MFCC features. After the first layer, the network includes a dropout layer with a rate of 0.3, which randomly sets 30% of the input units to zero in each iteration during training to prevent overfitting [39]. This dropout layer is followed by another dense layer with 512 units and a ReLU activation function. This pattern of a dense layer followed by a dropout layer is repeated three more times, with the number of units in each dense layer halving each time (256 units, 128 units, and 64 units, respectively). The final layer is a dense output layer with 10 units and a softmax activation function. This layer produces the final output of the network, which is a probability distribution over the 10 possible classes. This network architecture is more complex than the previous one, with more hidden layers and more units per layer, which allows it to potentially learn more complex and higher-level features from the input data. However, it is also more computationally expensive and may be more susceptible to overfitting without the use of dropout layers.

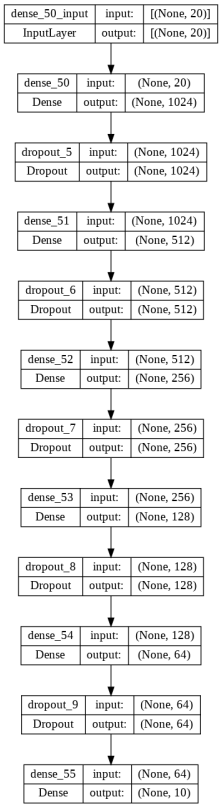

Fig. 9. NN Model Architecture with rmsProp

2) Recurrent Neural Network: A recurrent Neural Network (RNN) is a special kind of neural network suitable for various sequence labeling tasks [31]. In RNNs, each node has an additional input [32] computed from the activation of the layer in the previous time step. Therefore, RNN has the memory of the model, which makes up for the defect that the deep neural network (DNN) cannot capture the time dependence of the audio sequence and ignore the melody changes between different music. However, as new information flows into the network, the potential impact of previous time steps on the current output decreases rapidly, so it is difficult for RNNs to learn long-term dependencies, and there is a problem of gradient disappearance [32]. So, we use Long Short-Term Memory (LSTM) to avoid the non-linear problem in the recursion in RNN. Each LSTM cell takes input data as a time series, and at each time step, they decide whether to store, forget, or output the collected information [32]. And the output of each LSTM cell is another time series.

Our LSTM model has a total of 10 layers. The first layer is an input layer that accepts data. The the model contains two LSTM layers with the number of units being 128 and 32 respectively. We added a dropout of 0.05 and a recurrent dropout of 0.35 in the LSTM layer to avoid overfitting the model. Next is a Dense Layer with 32 units using the relu function as the activation function. We then added a Batch Normalization layer and a Dropout layer of 0.2 to prevent overfitting. This is followed by two Dense layers with 16 units and a relu activation function, with a 0.2 Dropout layer in between. Finally, the output layer uses the softmax function that outputs 10 different labels. The LSTM model has a total of 89356 variables of which the number of trainable variables is 89292. The architecture of this model has been illustrated in Figure 10.

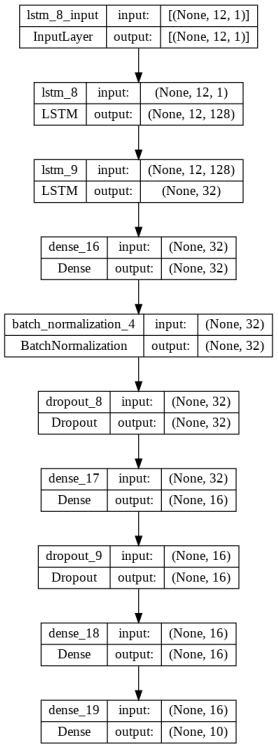

Fig. 10. LSTM Model Architecture

IV. Experimental Results

In this section, we empirically evaluate the performance of different models using original and augmented datasets. In section IV-B, we experiment with our deep neural network and evaluate the model. While in section IV-C, we present our findings from experiments on recurrent neural network.

A. Experimental Setup

Dataset. We experiment with the popular GTZAN dataset. As III-A describes, the dataset has a total of 1000 audio files uniformly distributed over 10 different genres. To assess how substantial audio dataset impacts overall performance, we augment the original data (described in III-D). The augmented dataset contains a total of 2997 audio files. We split the dataset on a 70/15/15 ratio for train/validation/test sets(both on original and augmented data).

Evaluation metrics We evaluate the performance of different deep-learning approaches using multiple metrics:

Accuracy. It is described as a ratio between the number of correct predictions vs. total number of predictions [33]. It can be defined as follows:

$$Accuracy = \frac{TN + TP}{TN + FP + TP + FN}$$

Here, TN = True Negative, TP = True Positive, FN = False Negative, TP = False Positive

F1-score. As this is a multi-class classification problem, we also evaluated F1-score. F1-score provides the harmonic mean of precision and recall. Formula of F1-score is as follows:

$$F1 - score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Confusion Matrix. Using this matrix we can easily visualize correct and incorrect predictions that a machine learning model generates relative to the actual class labels of the dataset. We create a 10*10 confusion matrix to visualize how our models perform while classifying those genres.

Model Parameters. Adam uses momentum and adaptive learning rate to speed up the convergence speed, which has the advantages of high computational efficiency, and low memory requirements, and is suitable for data and larger problems [34]. Therefore, we mostly choose Adam as the optimizer for both neural network and LSTM models to improve computational efficiency and improve training accuracy. We use the default learning rate, 0.001 for Adam optimizer. To assess the impact of learning rate and make it adaptive, we also make use of RMSProp optimizer. RMSProps facilitates adaptive learning rate through out the model training [35]. Categorical cross-entropy is suitable for multi-class classification, consisting of a Softmax activation plus a cross-entropy loss. Since our dataset has 10 different categories, we choose categorical cross-entropy as the loss function. We classify the data set into multiple categories. Thus, we do not use MeanSquaredError suitable for regression models or AUC suitable for binary classification as the evaluation index. We use accuracy as the evaluation index of the model.

B. Neural Network Results

We experiment with MFCC and Chroma STFT features dataset to assess the performance over different neural network models. Moreover, we inspect the impact of data augmentation on those models. Lastly, using accuracy, F1-score, and confusion matrix, we evaluate the models thoroughly.

1) Training Results for Neural Network Models: As we have a dataset that is not that big, we need to update different model parameters during the training process. We ensure that using 500 epochs and a batch size of 128. We test with a different number of epochs and batch sizes. The above-mentioned epochs guarantee that hyperparameters' weight changes over the training so that the learning curve goes from underfitting to optimal and would not reach overfitting [36]. Batch size confirms that better training will happen on a chunk of data instead of the whole dataset at once. For the model with RMSProp optimizer, we add dropout layers to avoid overfitting. Because of repeated rescaling in batch-normalization, exploding gradients can impact our neural networks adversely. That is why though we test using batch-normalization, we do not add those results here. For the first layer of the neural network, we pass MFCC data with input shape of 20 and 12 for Chroma STFT data.

Figure 11 to Figure 14 indicate training vs validation loss and accuracy for MFCC original data, MFCC augmented data, Chroma original data and Chroma augmented data respectively.

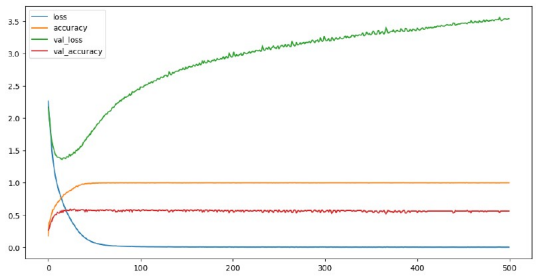

Fig. 11. Training result for NN model using the original MFCC features

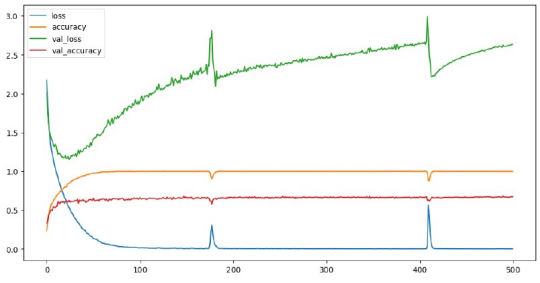

Fig. 12. Training result for NN model using the Augmented MFCC features

According to these plots, neural network models overfit in all cases. There can be multiple reasons behind that. Firstly, neural networks have the capability of learning million to

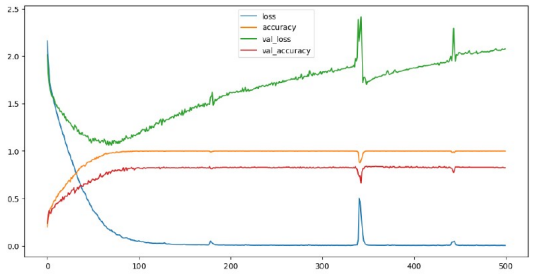

Fig. 13. Training result for NN model using the original Chroma STFT features

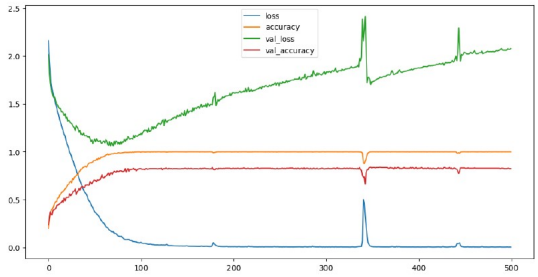

Fig. 14. Training result for NN model using the augmented Chroma STFT features

billion of parameters during model training [37], so it can even train on the noise in the audio signals. As our dataset may not be big enough for training on the neural network. Secondly, the audio preprocessing and normalization of the data may not be sufficient before passing into the neural network pipeline.

2) Evaluations for Neural Network Models: As stated, we evaluate neural network models using varying feature type(MFCC and Chroma STFT) and compare them based on original and augmented dataset. Accuracy results of these models is following in the tabular format:

TABLE I
NEURAL NETWORK MODEL EVALUATIONS

Sl	Feature Type	Accuracy	Feature set
1	MFCC (RMSProp)	57.99%	Original
2	MFCC	59.33%	Original
3	MFCC	69.77%	Augmented
4	Chroma STFT	84.33%	Original
5	Chroma STFT	64.99%	Augmented

Table I illustrates that though data augmentation can help improve accuracy in MFCC data, it does not have a similar impact on Chroma features. We verify these results with multiple rounds of experiments. One of the potential rationales can be not equally weighting the two coefficients. One possible solution to this can be combining these two features and experimenting neural network on the combined data. This can be a future direction to extend our work **??**.

Alongside accuracy, we evaluate F1-score for all the models to better assess the cost of False Positive and False Negative. This also confirms the distribution of the predicted genres. Figure 15 to Figure 18 represents F1-score for models 2-5 from Table I .

	precision	recall	f1-score	support
0	0.71	0.83	0.77	12
1	1.00	0.83	0.91	18
2	0.33	0.30	0.32	10
3	0.33	0.38	0.36	13
4	0.44	0.50	0.47	16
5	0.58	0.88	0.70	17
6	0.63	0.75	0.69	16
7	0.75	0.50	0.60	18
8	0.67	0.57	0.62	14
9	0.40	0.25	0.31	16
accuracy			0.59	150
macro avg	0.59	0.58	0.57	150
weighted avg	0.60	0.59	0.59	150

Fig. 15. F1-score for NN model using the original MFCC features

	precision	recall	f1-score	support
0	0.58	0.68	0.63	31
1	0.89	0.91	0.90	55
2	0.58	0.57	0.57	44
3	0.67	0.51	0.58	43
4	0.69	0.67	0.68	46
5	0.67	0.74	0.71	47
6	0.85	0.81	0.83	43
7	0.78	0.90	0.84	42
8	0.61	0.62	0.62	48
9	0.59	0.53	0.56	51

Fig. 16. F1-score for NN model using the Augmented MFCC features

	precision	recall	f1-score	support
0	0.77	0.71	0.74	38
1	0.94	1.00	0.97	30
2	0.78	0.78	0.78	23
3	0.85	0.85	0.85	27
4	0.85	0.88	0.87	33
5	0.89	0.92	0.91	26
6	0.88	0.85	0.87	27
7	0.83	0.91	0.87	33
8	0.86	0.83	0.85	30
9	0.77	0.73	0.75	33

Fig. 17. F1-score for NN model using the original Chroma STFT features

Here, precision refers to the correct predictions with respect to total positive predictions while recall specifies correct positive predictions with respect to total actual positives. Comparing presicion, recall, F1-score of these models again validate

	precision	recall	f1-score	support
0	0.68	0.68	0.68	31
1	0.57	0.57	0.57	23
2	0.70	0.70	0.70	27
3	0.61	0.66	0.63	29
4	0.76	0.56	0.65	39
5	0.66	0.79	0.72	29
6	0.62	0.75	0.68	32
7	0.67	0.67	0.67	24
8	0.47	0.64	0.54	25
9	0.81	0.54	0.65	41

Fig. 18. F1-score for NN model using the augmented Chroma STFT features

our earlier finding that data augmentation helps improving performance in MFCC dataset but not for Chroma data.

To visualize how actually the audio instances are predicted by our models, we also check the confusion matrix (Figure 19 to Figure 22).

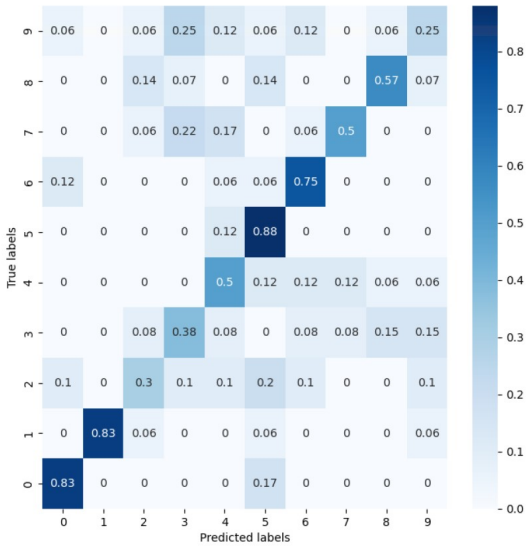

Fig. 19. Confusion matrix for NN model using the original MFCC features

The confusion matrix depicts that most of the models correctly predict *classical* audio files while giving poor performance while classifying *country* audio files. This can be due to our choice of features of MFCC and Chroma. Provided that we choose other features(e.g. zero-crossing, pitch, spectral entropy, etc.) correct predictions might change [40].

C. Recurrent Neural Network Results

We sequentially use Chroma and MFCC features to test the performance of the LSTM model on different features. At the same time, to test the impact of data augmentation on the results, we test the performance of the LSTM model before and after data augmentation on the test set. We use accuracy and confusion matrix as evaluation metrics.

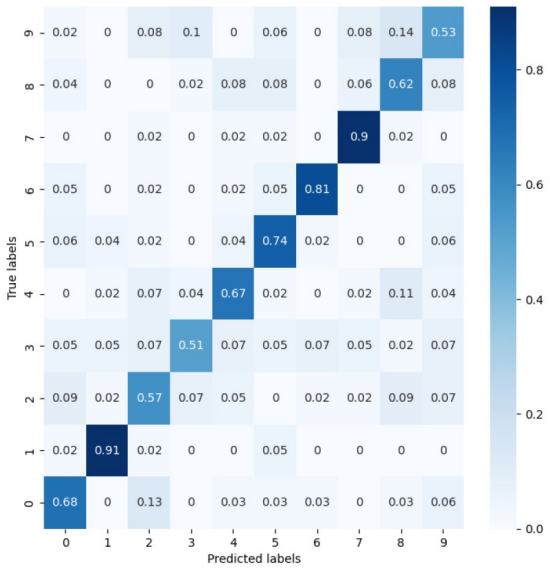

Fig. 20. Confusion matrix for NN model using the Augmented MFCC features

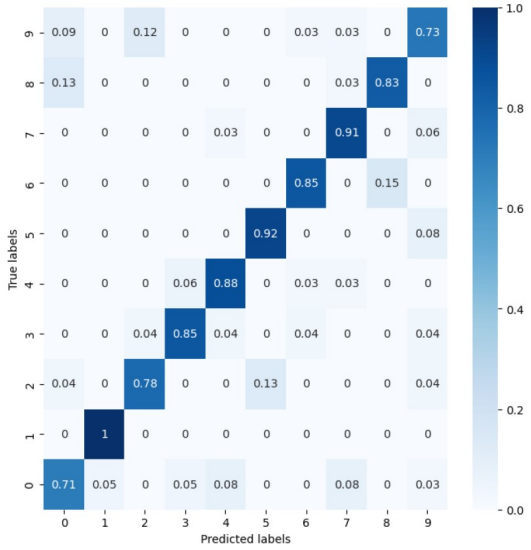

Fig. 21. Confusion matrix for NN model using the original Chroma STFT features

1) Training Results for LSTM Model: In training, we use 500 epochs and a batch size of 16. This is because we want our LSTM model to be trained enough to bring the training and validation loss down and stable. We also add BatchNormalization with each batch's mean and standard deviation to normalize the values of the units and Dropout layers to randomly select neurons and set their weights to zero to avoid overfitting. The batch size of 16 is chosen because the size of our data set is smaller than other data sets suitable for RNN model training, so we choose a lower batch size to ensure the accuracy of training. Figures 23 to 26 show the training and validation accuracy and loss of the LSTM model for 500 epochs of training. Figure 23 shows the LSTM

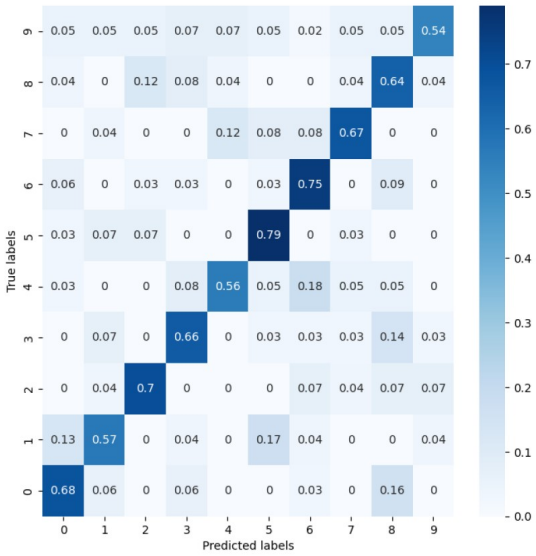

Fig. 22. Confusion matrix for NN model using the augmented Chroma STFT features

model using the original MFCC features, Figure 24 shows the LSTM model using the augmented MFCC features, Figure 25 shows the LSTM model using the original Chroma features, and Figure 26 shows the LSTM model using the augmented Chroma features.

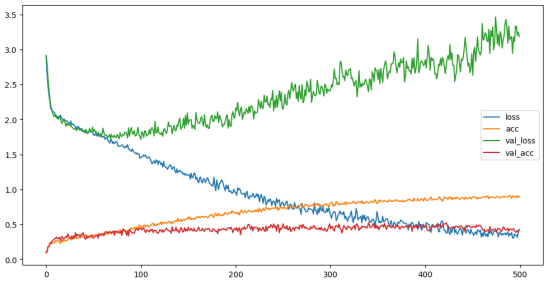

Fig. 23. Training Result for LSTM model Using the Original MFCC Features

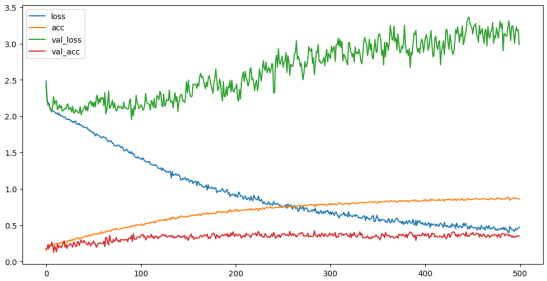

Fig. 24. Training Result for LSTM model Using the Augmented MFCC Features

We can conclude that the LSTM model has overfitting except in the original Chroma features. This could be because the data sample size is not enough, or the LSTM model is too complex.

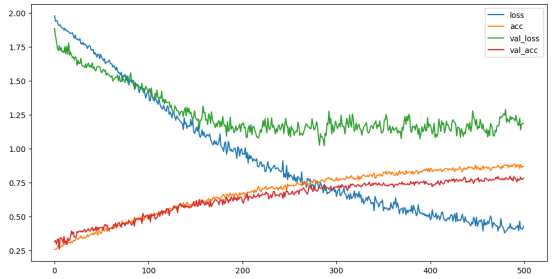

Fig. 25. Training Result for LSTM model Using the Original Chroma Features

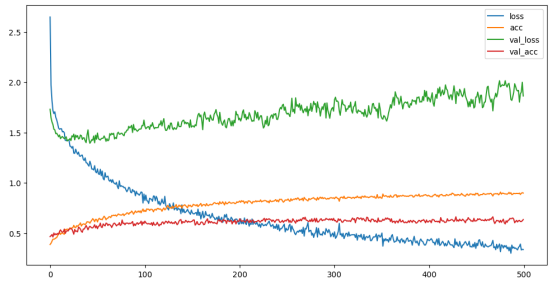

Fig. 26. Training Result for LSTM model Using the Augmented Chroma Features

2) Evaluations for LSTM Model: We test the performance of the LSTM model in different features on the test set, and the results are shown in Table II.

TABLE II
LSTM MODEL EVALUATIONS

Feature Type	Dataset	Accuracy
MFCC	Original	40.67%
MFCC	Augmented	57.33%
Chroma STFT	Original	60.67%
Chroma STFT	Augmented	65.00%

From the table, we can conclude that data augmentation helps improve the test accuracy of the LSTM model both in the case of MFCC and Chroma features. Therefore, we conclude that data augmentation can help the LSTM model better distinguish music types and improve classification accuracy.

In addition to the accuracy of the test set, we evaluate the LSTM model using the F1-score. F1-score can help balance precision and recall, that is, balance positive and negative samples. To test the performance of the LSTM model in dealing with imbalanced data sets, we calculate the F1-score value of the LSTM model in different data through the confusion matrix. Figures 27 to 30 describe the F1-score analysis of the LSTM model in different cases. Figure 27 is the result using the original MFCC features, Figure 28 is the F1-score of the augmented MFCC features, Figure 29 shows the result using the original Chroma features, and Figure 30 shows the F1-score using the augmented Chroma features.

By comparing Figure 27 and 28 and comparing Figures 29 and 30, we find that no matter whether it is MFCC feature

	precision	recall	f1-score	support
0	0.36	0.42	0.38	12
1	0.80	0.44	0.57	18
2	0.27	0.30	0.29	10
3	0.27	0.46	0.34	13
4	0.56	0.62	0.59	16
5	0.62	0.59	0.61	17
6	0.78	0.44	0.56	16
7	0.38	0.56	0.45	18
8	0.33	0.07	0.12	14
9	0.05	0.06	0.05	16
accuracy			0.41	150
macro avg	0.44	0.40	0.40	150
weighted avg	0.46	0.41	0.41	150

Fig. 27. F1 Results for LSTM model Using the Original MFCC Features

	precision	recall	f1-score	support
0	0.62	0.52	0.56	31
1	0.90	0.85	0.88	55
2	0.44	0.45	0.45	44
3	0.45	0.47	0.46	43
4	0.60	0.59	0.59	46
5	0.57	0.66	0.61	47
6	0.70	0.65	0.67	43
7	0.60	0.62	0.61	42
8	0.49	0.58	0.53	48
9	0.34	0.29	0.32	51
accuracy			0.57	450
macro avg	0.57	0.57	0.57	450
weighted avg	0.58	0.57	0.57	450

Fig. 28. F1 Results for LSTM model Using the Augmented MFCC Features

or Chroma feature, after data augmentation, the F1-score of the LSTM model has been improved. Therefore, we can conclude that data augmentation can not only help improve the test accuracy of LSTM model but also help improve the performance of the model in imbalanced datasets.

The confusion matrix can observe the performance of the model on different categories and is used to calculate the precision and recall of the model corresponding to different categories. The confusion matrix results of the LSTM model in different experimental tests are shown in Figure 31 to Figure 34.

The confusion matrix of the LSTM model has poor prediction results in the original MFCC features and has high accuracy in predicting *classical* and *jazz* audio files in the augmented MFCC features and Chroma features. However, the model generally performs poorly on classifying *rock* audio files in four different cases. Overall, the LSTM model has the

	precision	recall	f1-score	support
0	0.86	0.63	0.73	38
1	0.64	0.83	0.72	30
2	0.61	0.74	0.67	23
3	0.40	0.59	0.48	27
4	0.55	0.55	0.55	33
5	0.63	0.65	0.64	26
6	0.60	0.56	0.58	27
7	0.61	0.58	0.59	33
8	0.52	0.50	0.51	30
9	0.80	0.48	0.60	33
accuracy			0.61	300
macro avg	0.62	0.61	0.61	300
weighted avg	0.63	0.61	0.61	300

Fig. 29. F1 Results for LSTM model Using the Original Chroma Features

	precision	recall	f1-score	support
0	0.72	0.74	0.73	31
1	0.53	0.70	0.60	23
2	0.49	0.70	0.58	27
3	0.77	0.59	0.67	29
4	0.76	0.56	0.65	39
5	0.73	0.83	0.77	29
6	0.70	0.66	0.68	32
7	0.54	0.58	0.56	24
8	0.66	0.76	0.70	25
9	0.67	0.49	0.56	41
accuracy			0.65	300
macro avg	0.66	0.66	0.65	300
weighted avg	0.67	0.65	0.65	300

Fig. 30. F1 Results for LSTM model Using the Augmented Chroma Features

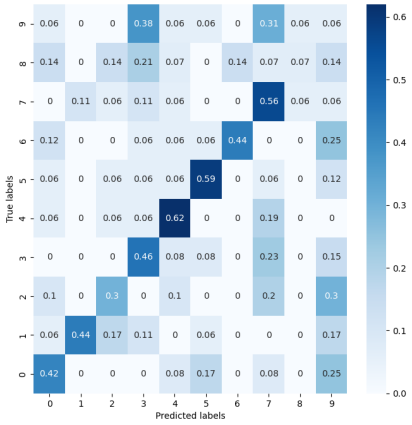

Fig. 31. Confusion Matrix for LSTM model Using the Original MFCC Features

best prediction accuracy for the augmented Chroma features. Therefore, we can conclude that the Chroma feature is more

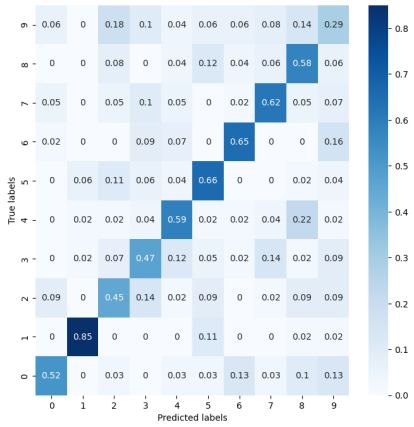

Fig. 32. Confusion Matrix for LSTM model Using the Augmented MFCC Features

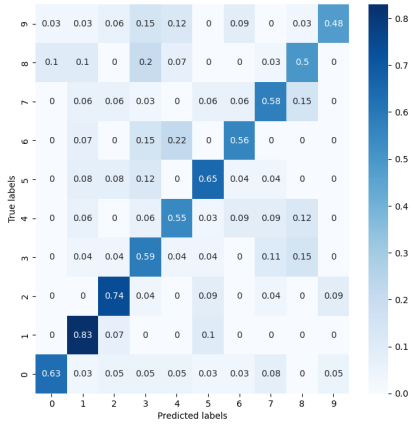

Fig. 33. Confusion Matrix for LSTM model Using the Original Chroma Features

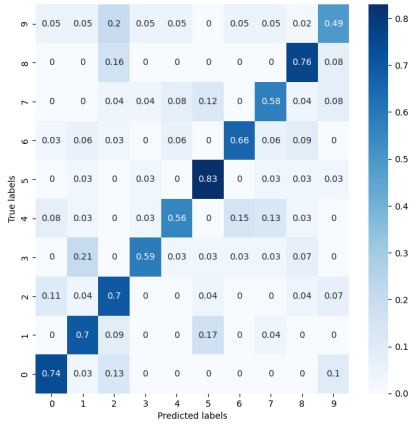

Fig. 34. Confusion Matrix for LSTM model Using the Augmented Chroma Features

suitable for the LSTM model and that data augmentation is helpful for the accurate classification of the model.

In conclusion, data augmentation is a powerful improvement to the LSTM model. Data augmentation can help the model to train better to improve classification accuracy. At the same time, data augmentation can also help the model adapt to the imbalanced data set by creating more data set in the training set so that the model can learn the feature of each particular genre or class more accurately which can help in detecting the unseen data set.

V. Conclusion and Future Work

A crucial feature for any music streaming platform is the ability to rapidly categorise songs in any playlist or library by genre. To better understand the music information retrieval process, we have evaluated some prior machine learning and deep learning researches on music and genre categorization on this paper. We have implemented two different Neural Network to classify the music genre with two distinct and dominating spectral features in the audio classification. Additionally, data scarcity is a major drawback while training any neural network and we have implemented two data augmentation technique on the GTZAN dataset to increase the training dataset. The experimental results indicate that both the model yield comparatively better result when it is trained with the augmented data and MFCC. This simply outlines that dataset created through data augmentation are valuable because it helps in increasing the predicted accuracy and overall performance of neural network models by lowering the risk of overfitting, which occurs when algorithms detect erroneous values in a dataset.

With respect to the model architecture, some future work may include removing the very last layer of both NN and RNN to see what probabilities both the models can computer before converting to a one-hot array of values. It will also entail modifying the optimizer function, weight initialization, and the activation function to better understand the impact of each of these changes. Secondly, stacking of different feature channels and using the stacked feature as an input to the neural network classifiers can add significant dimension in classification as the classifier will be able to learn features from two distinct spectral. Furthermore, upgrading our model with Residual Learning, which has been found to significantly enhance classification tasks [41], can be an option to look forward which might aid in producing better results across a larger number of classes.

References

[1] “Are Music genres important? 3 ways they improve our experience with music,” Preston Cram, 03-May-2022. [Online]. Available: https://www.prestoncram.com/post/are-music-genres-important-3-ways-they-enhance-our-enjoyment-of-music%20to%20appreciate%20quality. [Accessed: 27-Nov-2022].

[2] G. Tzanetakis and P. Cook, ”Musical genre classification of audio signals,” in IEEE Transactions on Speech and Audio Processing, vol. 10, no. 5, pp. 293-302, July 2002, doi: 10.1109/TSA.2002.800560.

[3] H. Yuan, W. Zheng, Y. Song and Y. Zhao, ”Parallel Deep Neural Networks for Musical Genre Classification: A Case Study,” 2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC), 2021, pp. 1032-1035, doi: 10.1109/COMPSAC51774.2021.00140.

[4] S. Vishnupriya and K. Meenakshi, "Automatic Music Genre Classification using Convolution Neural Network," 2018 International Conference on Computer Communication and Informatics (ICCCI), 2018, pp. 1-4, doi: 10.1109/ICCCI.2018.8441340.

[5] Y. KIKUCHI, N. AOKI and Y. DOBASHI, "A Study on Automatic Music Genre Classification Based on the Summarization of Music Data," 2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), 2020, pp. 705-708, doi: 10.1109/ICAIIC48513.2020.9065046.

[6] B. L. Sturm, "The GTZAN dataset: Its contents, its faults, their effects on evaluation, and its future use," 2013, arXiv:1306.1461. [Online]. Available: http://arxiv.org/abs/1306.1461

[7] Tao Li and G. Tzanetakis, "Factors in automatic musical genre classification of audio signals," 2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (IEEE Cat. No.03TH8684), 2003, pp. 143-146, doi: 10.1109/ASPAA.2003.1285840.

[8] C. N. Silla Jr., A. L. Koerich, and C. A. Kaestner, "A machine learning approach to automatic music genre classification," Journal of the Brazilian Computer Society, vol. 14, no. 3, pp. 7–18, 2008.

[9] Y. Panagakis and C. Kotropoulos, "Music genre classification via topology preserving non-negative tensor factorization and sparse representations," 2010 IEEE International Conference on Acoustics, Speech and Signal Processing, 2010.

[10] S. Sigtia and S. Dixon, "Improved music feature learning with deep neural networks," 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2014, pp. 6959-6963, doi: 10.1109/ICASSP.2014.6854949.

[11] Li et al. Automatic musical pattern feature extraction using convolutional neural network. Lecture Notes in Engineering and Computer Science, 2180(1), 2010.

[12] Y. Xu and W. Zhou, "A deep music genres classification model based on CNN with Squeeze & Excitation Block," 2020 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), 2020, pp. 332-338.

[13] D. Bisharad and R. H. Laskar, "Music Genre Recognition Using Residual Neural Networks," TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON), 2019, pp. 2063-2068, doi: 10.1109/TENCON.2019.8929406.

[14] A. Olteanu, "GTZAN dataset - music genre classification," Kaggle, 24-Mar-2020. [Online]. Available: https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification.

[15] A. Elbir and N. Aydin, "Music genre classification and music recommendation by using Deep Learning," Electronics Letters, vol. 56, no. 12, pp. 627–629, 2020.

[16] B. Riggs, "Beginner Guide to Visualizing Audio as a spectrogram in Python," Dolby.io, 10-Oct-2022. [Online]. Available: https://dolby.io/blog/beginners-guide-to-visualizing-audio-as-a-spectogram-in-python/%20we%20wish%20to%20interpret. [Accessed: 01-Dec-2022].

[17] Q. Chen, L. Ai, W. Wei, W. Ma, P. Xie and M. Zhang, "Analysis of Music Representations of Vocal Performance Based on Spectrogram," 2010 6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM), 2010, pp. 1-4, doi: 10.1109/WICOM.2010.5600580.

[18] J. K. Das, A. Ghosh, A. K. Pal, S. Dutta and A. Chakrabarty, "Urban Sound Classification Using Convolutional Neural Network and Long Short Term Memory Based on Multiple Features," 2020 Fourth International Conference On Intelligent Computing in Data Sciences (ICDS), 2020, pp. 1-9, doi: 10.1109/ICDS50568.2020.9268723.

[19] "Audio data analysis using Deep Learning with python (part 1)," KDnuggets. [Online]. Available: https://www.kdnuggets.com/2020/02/audio-data-analysis-deep-learning-python-part-1.html. [Accessed: 01-Dec-2022].

[20] I. Kamarulafizam, S.-H. Salleh, J. M. Najeb, A. K. Ariff, and A. Chowdhury, "Heart sound analysis using MFCC and Time Frequency Distribution," 3rd Kuala Lumpur International Conference on Biomedical Engineering 2006, pp. 402–405, 2007.

[21] "Librosa.feature.mfcc," librosa.feature.mfcc - librosa 0.10.0.dev0 documentation. [Online]. Available: https://librosa.org/doc/main/generated/librosa.feature.mfcc.html. [Accessed: 01-Dec-2022].

[22] "LIBROSA.FEATURE.CHROMA_STFT," librosa.feature.chroma_stft - librosa 0.10.0.dev0 documentation. [Online]. Available: https://librosa.org/doc/main/generated/librosa.feature.chroma_stft.html.

[23] "Why we take only 12-13 mfcc coefficients in feature extraction?" [Online]. Available: https://www.researchgate.net/post/Why_we_take_only_12-13_MFCC_coefficients_in_feature_extraction.

[24] A. Mikołajczyk and M. Grochowski, "Data augmentation for improving deep learning in image classification problem," 2018 International Interdisciplinary PhD Workshop (IIPhDW), 2018, pp. 117-122, doi: 10.1109/IIPHDW.2018.8388338.

[25] K. Paralkar, "Audio data augmentation in python," Medium, 15-Apr-2020. [Online]. Available: https://medium.com/@keur.plkar/audio-data-augmentation-in-python-a91600613e47. [Accessed: 01-Dec-2022].

[26] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for Deep Learning," Journal of Big Data, vol. 6, no. 1, 2019.

[27] "5 techniques to prevent overfitting in neural networks," KDnuggets. [Online]. Available: https://www.kdnuggets.com/2019/12/5-techniques-prevent-overfitting-neural-networks.html: :text=As%20we%20can%20see%2C%20using,and%20is%20forced%20to%20generalize. [Accessed: 04-Dec-2022].

[28] "Gaussian noise - linear filtering," GIASSA.NET — Engineering, DIY, and Everything Else, 05-Apr-2015. [Online]. Available: https://www.giassa.net/?page_id=637: :text=A%20normalized%20Gaussian%20distribution%20has,values%20are%20from%20this%20point. [Accessed: 04-Dec-2022].

[29] "Lecture11: White and red noise ." [Online]. Available: https://atmos.washington.edu/ breth/classes/AM582/lect/lect8-notes.pdf. [Accessed: 04-Dec-2022].

[30] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016

[31] J. Dai, S. Liang, W. Xue, C. Ni, and W. Liu, "Long short-term memory recurrent neural network based segment features for music genre classification," 2016 10th International Symposium on Chinese Spoken Language Processing (ISCSLP), 2016.

[32] D. de Benito-Gorron, A. Lozano-Diez, D. T. Toledano, and J. Gonzalez-Rodriguez, "Exploring convolutional, recurrent, and hybrid deep neural networks for speech and music detection in a large audio dataset," EURASIP Journal on Audio, Speech, and Music Processing, vol. 2019, no. 1, 2019.

[33] "Evaluating a machine learning model." [Online]. Available: https://www.jeremyjordan.me/evaluating-a-machine-learning-model/. [Accessed: 06-Dec-2022].

[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv.org, 30-Jan-2017. [Online]. Available: https://arxiv.org/abs/1412.6980. [Accessed: 07-Dec-2022].

[35] "A Complete Guide to Adam and RMSprop Optimizer." [Online]. Available: https://medium.com/analytics-vidhya/a-complete-guide-to-adam-and-rmsprop-optimizer-75f4502d83be. [Accessed: 07-Dec-2022].

[36] "Epoch vs Batch Size vs Iterationsr." [Online]. Available: https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9. [Accessed: 07-Dec-2022].

[37] "Complete Guide to Prevent Overfitting in Neural Networks." [Online]. Available: https://www.analyticsvidhya.com/blog/2021/06/complete-guide-to-prevent-overfitting-in-neural-networks-part-1/: :text=Why

[38] "Classifying music audio with timbral and chroma features" [Online]. Available: https://www.ee.columbia.edu/ dpwe/talks/timbrechroma-poster-07.pdf. [Accessed: 06-Dec-2022].

[39] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." The journal of machine learning research 15.1 (2014): 1929-1958.

[40] "What features to consider while training audio files?" [Online]. Available: https://towardsdatascience.com/how-i-understood-what-features-to-consider-while-training-audio-files-eedfb6e9002b. [Accessed: 05-Dec-2022].

[41] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.