Music Genre Classification

Rakesh Roshan Paul — *170104130@aust.edu*
Wasif Mohammad — *170104139@aust.edu*
Faysal Ahmed — *170104137@aust.edu*
Mallik Galib Shahriar — *160204079@aust.edu*

I. Introduction

We've all listened to music on a streaming app. Music Genre Classification System is one way to demonstrate reasoning. Classifying music based on genre is a necessary first step in creating an effective recommendation system. Sound files will be handled in Python, audio features will be calculated from them, Machine Learning Algorithms will be used to see the outcomes, and so on. The main goal is to construct a machine learning model that categorizes music samples into distinct genres in a more systematic manner. Its goal is to forecast the genre based on the input, which is an audio file. The goal of automating music classification is to make it easier and faster to choose tunes. Manually classifying songs or music necessitates listening to a large number of tracks before deciding on a category. This takes a lot of time and effort. By automating the categorizing of music, useful information like trends, popular genres, and performers can be found more quickly and conveniently Knowing what types of music you like is the first step towards figuring this out. In our project, we have implemented Music Genre Classification by using Long Short Term Memory(LSTM). For this project, the dataset that we will be working with is GTZAN Genre Classification dataset which consists of 1,000 audio tracks, each 30 seconds long[2]. It contains 10 genres, each represented by 100 tracks.Also we showed performance comparison by tuning hyper parameters to increase the accuracy. We were able to train a model that can classify music from 10 different genres by using LSTM. Our task was challenging because,in first our model gave the accuracy 30%. Our main challenge was to increase the training accuracy.

II. Related Works

The authors et al. [3] presented a music genre recognition using spectograms. They worked on music genre classification using different types of inputs including spectrograms. To analysis performance the authors et al.[3] used 900 songs from Latin Music Database. They divided them into 10 music genre equally. From each of the 900 songs they extracted 30 s segements, these were represented as ten 28-dimensional feature vectors. The authors et al.[4] presented a method for the selection of training instances based Support Vector Machine(SVM) classifier. By this they got the accuracy of 60%.The authors et al.[5] implemented a method to train the dataset by using Long Short Term Memory(LSTM). Their first approach was train the dataset on 6 genre. In the second approach, they adopt a hierarchical divide and conquer strategy to achieve 10 genres classification. When they applied on two genres their model testing accuracy was 98.15% but when they increased the music genres, their testing accuracy decreased. By using 4 genres they got 51.88%.

III. Project Objectives

The GTZAN dataset is the most used public dataset for evaluation in machine listening ressearch for music genre recognition (MGR). The files were collected in 2000-2001 from avariety of sources including personal CDs, radio, microphone recordings, in order to represent a variety of recording conditions.

	filename	length	chroma_stft_mean	chroma_stft_var	rms_mean	rms_var	spectral_centroid_mean	spectral_centroid_var	spectral_bandwidth_mean	spectral_bandwidth_var	rolloff_mean	rolloff_var	zero_crossing_rate_mean	zero_crossing_rate_var	harmony_mean	harmony_var
0	blues.00000.0.wav	66149	0.335406	0.091048	0.130405	0.003521	1773.065032	167541.630869	1972.744388	117335.771563	3714.560359	1.080790e+06	0.081851	0.000558	-0.000078	0.008354
1	blues.00000.1.wav	66149	0.343065	0.086147	0.112699	0.001450	1816.693777	90525.690866	2010.051501	65671.875673	3869.682242	6.722448e+05	0.087173	0.001030	-0.000099	0.004950
2	blues.00000.2.wav	66149	0.346815	0.092243	0.132003	0.004620	1788.539719	111407.437613	2084.565132	75124.921716	3997.639160	7.907127e+05	0.071383	0.000425	-0.000066	0.012476
3	blues.00000.3.wav	66149	0.363639	0.086856	0.132565	0.002448	1655.289045	111952.284517	1960.039988	82913.639269	3568.300218	9.216524e+05	0.069426	0.000304	-0.000014	0.008318
4	blues.00000.4.wav	66149	0.335579	0.088129	0.143289	0.001701	1630.656199	79667.267654	1948.503884	60204.020268	3469.992864	6.102111e+05	0.070095	0.000289	0.000041	0.009634
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9985	rock.00099.5.wav	66149	0.349126	0.080515	0.050019	0.000097	1499.083005	164266.886443	1718.707215	85931.574523	3015.559458	8.479527e+05	0.072885	0.000661	-0.000034	0.002113
9986	rock.00099.6.wav	66149	0.372564	0.082626	0.057897	0.000088	1847.965128	281054.935973	1906.468492	99727.037054	3746.694524	1.170890e+06	0.089111	0.000987	-0.000027	0.002897
9987	rock.00099.7.wav	66149	0.347481	0.089019	0.052403	0.000701	1346.157659	662956.246325	1561.859087	138762.841945	2442.362154	2.602871e+06	0.072194	0.003784	0.000015	0.002455
9988	rock.00099.8.wav	66149	0.387527	0.084815	0.066430	0.000320	2084.515327	203891.039161	2018.366254	22860.992562	4313.266226	4.968878e+05	0.104072	0.001951	0.000004	0.003681
9989	rock.00099.9.wav	66149	0.369293	0.086759	0.050524	0.000067	1634.330126	411429.169769	1867.422378	119722.211518	3462.042142	1.517016e+06	0.068495	0.001388	-0.000032	0.002151

9990 rows × 60 columns

Fig. 1. Dataset

In the dataset the 'filename' feature column and the class label are categorical data. We used label encoding to encode the categorical values to numerical values. After applying Label encoder:

```
#Unique value for Label
le = LabelEncoder()
df['label'] = le.fit_transform(df['label'])
df['label'].unique()
```

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

Fig. 2. Label Encoder

We'll drop the first column 'filename' as it is unnecessary because it is a categorical value. Standard scaler is used to

standardize features by removing the mean and scaling to unit variance.Standardization of a dataset is a common requirement for many machine learning estimators: they might behave badly if the individual features do not more or less look like standard normally distributed data.

We have showed a sample output for prediction.

```
make_prediction(model, X_test, y_test,9)

---Now testing the model for one audio file---
The model predicts: rock, and ground truth is: rock.
```

Fig. 3. Prediction

IV. Methodology

In our proposed Methodology, we have first collected the audio files from Gtzan Music Dataset[2]. From the dataset, we tried to understand the audio files.After that, we tried to visualized the audio files. After doing so, we applied feature extraction process,because,model can not run text values for that we converted it to numerical values. Then, we trained our model by using LSTM. Finally, we have evaluated the performance by using Confusion Matrix, also calculated the Precision, Recall and F1-score.

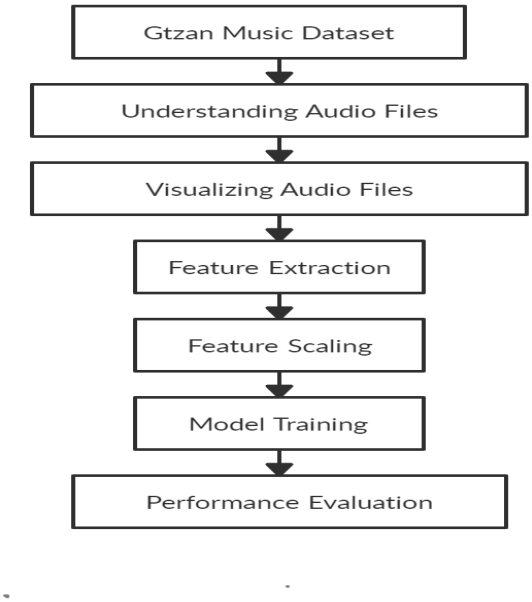

Fig. 4. Flow diagram of the whole process.

A. Understanding Audio Files

Librosa is a python package for music and audio analysis. It provides the building blocks necessary to create music information retrieval systems. By using Librosa, we can extract certain key features from the audio samples such as Tempo, Chroma Energy Normalized, Mel-Freqency Cepstral Coefficients, Spectral Centroid, Spectral Contrast, Spectral Rolloff and Zero Crossing Rate. For that, we used Librosa to understand audio files type, by taking sample files from the dataset[2].

B. Visualizing Audio Files

Waveforms are visual representations of sound as time on the x-axis and amplitude on the y-axis. They are great for allowing us to quickly scan the audio data and visually compare and contrast which genres might be more similar than others.

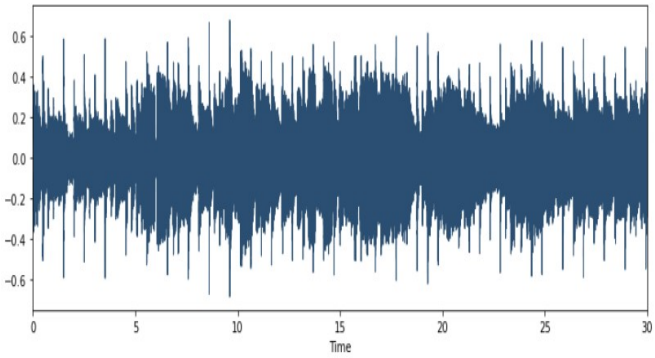

Fig. 5. Raw Waves of Samples.

C. Feature Extraction

Preprocessing of data is required before we finally train the data. We will try and focus on the last column that is 'label' and will encode it with the function LabelEncoder() of sklearn.preprocessing.

```
class_list=df.iloc[:,-1]
convertor=LabelEncoder()

df.iloc[:,-1]

0       blues
1       blues
2       blues
3       blues
4       blues
        ...
9985     rock
9986     rock
9987     rock
9988     rock
9989     rock
Name: label, Length: 9990, dtype: object

y=convertor.fit_transform(class_list)
y

array([0, 0, 0, ..., 9, 9, 9])
```

Fig. 6. Feature Extracting Code Segment.

We can't have text in our data if we're going to run any kind of model on it. So before we can run a model, we need to make this data ready for the model. To convert this kind of categorical text data into model-understandable numerical data, we use the Label Encoder class.

D. Feature Scaling

Standard scaler is used to standardize features by removing the mean and scaling to unit variance.Standardization of a dataset is a common requirement for many machine learning estimators: they might behave badly if the individual features do not more or less look like standard normally distributed data.

E. Model Training

Now comes the last part of the music classification genre project. The features have been extracted from the raw data and now we have to train the model. There are many ways through which we can train our model. Some of these approaches are: Multi class Support Vector Machines, K-Means Clustering, K-Means Neighbors, Convolutional Neural Networks, Recurrent Neural Network. For this project, we will be using Long Short Term Memory. Algorithm for training our model. We chose this approach because various forms of research show it to have the best results for this problem. For the LSTM model, we had used the Adam optimizer for training the model. The epoch that was chosen for the training model is 600. All of the hidden layers are using the Softmax activation function and the output layer uses the softmax function. The loss is calculated using the sparse-categorical-crossentropy function. Dropout is used to prevent overfitting. We chose the Adam optimizer because it gave us the best results after evaluating other optimizers.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_60 (LSTM)               (None, 58, 256)           264192
_________________________________________________________________
lstm_61 (LSTM)               (None, 58, 128)           197120
_________________________________________________________________
lstm_62 (LSTM)               (None, 64)                49408
_________________________________________________________________
dense_31 (Dense)             (None, 10)                650
=================================================================
Total params: 511,370
Trainable params: 511,370
Non-trainable params: 0
_________________________________________________________________
None
```

Fig. 7. LSTM Model Architecture.

F. Performance Evaluation

After getting the training accuracy,testing accuracy in our dataset by using LSTM , we have also used a machine learning algorithms like Linear Support Vector Machine (SVM), Decision Tree, Logistic Regression, Naive-Bayes, KNN (k-Nearest Neighbors), RandomForest classifier and Stochastic Gradient Descent (SGD) for performance comparison. After training these machine learning algorithms, we have done the performance evaluation test on our test dataset to evaluate our LSTM trained model.

V. Experiments

A. Dataset

Classification dataset which consists of 1,000 audio tracks, each 30 seconds long. It contains 10 genres, each represented by 100 tracks. The Genres are: Blues, Classical, Disco, Hip-hop, Jazz, Metal, Pop, Reggae, Rock. All the levels are equally distributed, here we can see a pie chart from the above discussion.

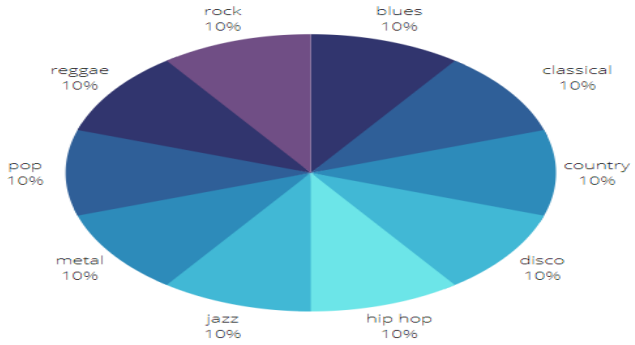

Fig. 8. Distribution of Music Genres.

B. Evaluation Metrics

In this section, we will discuss about the Performance evaluation of our LSTM trained model. Also we will show how we increased our training and testing accuracy by tuning. We will give a short description in this section.
After applying epoch= 7, batch size= 128,learning rate= 1e-2 as hyperperameter. We get,

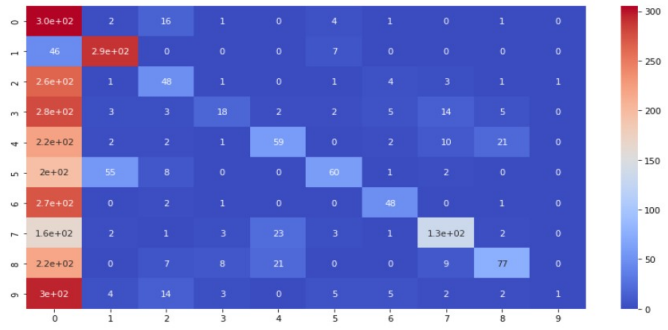

Fig. 9. Confusion Matrix of LSTM Model.

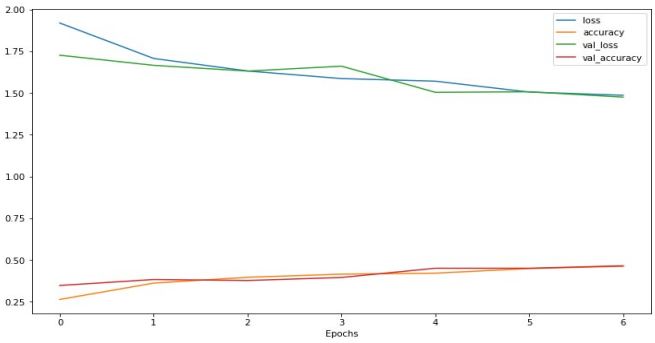

Fig. 10. Loss Curve of LSTM Model.

After applying epoch= 40, batch size= 220,learning rate= 1e-1 as hyperparameter.We get,

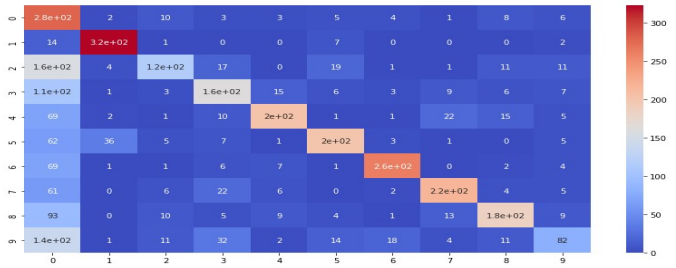

Fig. 11. Confusion Matrix of LSTM Model.

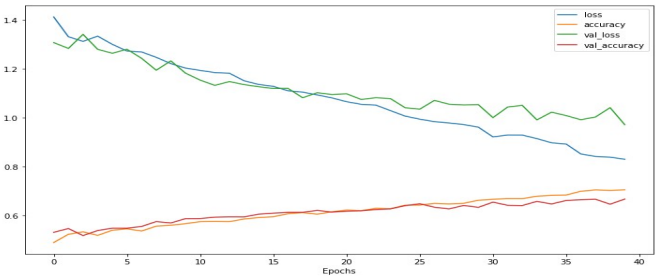

Fig. 12. Loss Curve of LSTM Model.

After applying epoch= 50, batch size= 220 ,learning rate= 1e-1 as hyperparameter. We get,

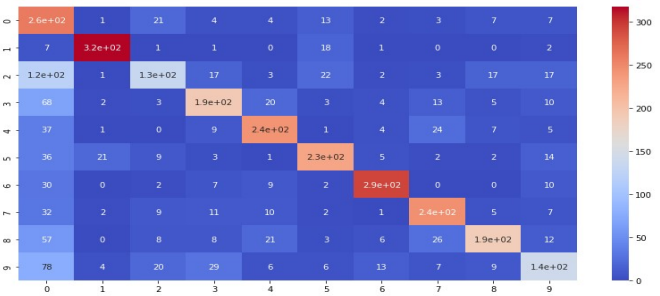

Fig. 13. Confusion Matrix of LSTM Model

In the above we showed how we increased our training and testing accuracy by hyperparameter tuning.

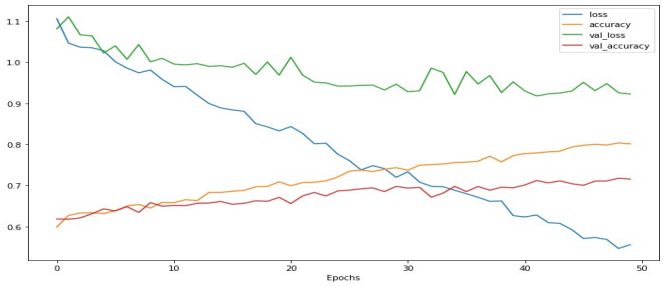

Fig. 14. Loss Curve of LSTM Model

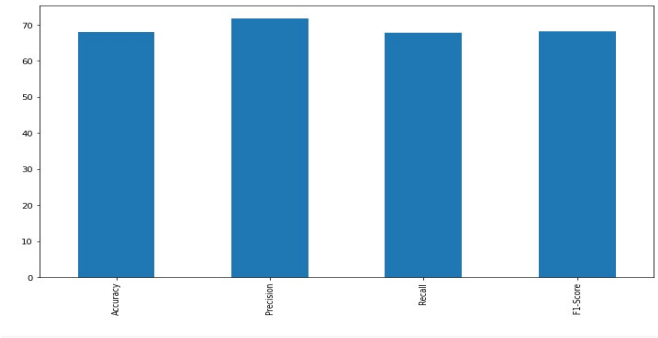

Fig. 15. Chart of Precision, Recall, F1-Score

C. Result

In this section, we have shown experiment comparisons of different experiments that we made. Here, our first experiment started with epoch= 7, batch size= 128, learning rate= 1e-2. Here we got the accuracy of 0.4626, in the Experiment2 we used epoch with 20 and having same batch size and learning rate. Again we used same epoch and learning rate but this time we used batch size=220. Then we get the accuracy of 0.5867. Finally we used same batch size but increased the learning rate as 1e-1 and also increased epoch number as 50. After training the model we got accuracy of 0.8014 or 80.14%.

TABLE I
PERFORMANCE COMPARISON USING LSTM MODEL

Input	Precision	Recall	F1-Score	Accuracy
Experiment1	58.98	31.24	30.83	0.4626
Experiment2	64.45	48.63	49.24	0.5867
Experiment3	67.99	50.83	52.07	0.5306
Experiment4	71.32	61.34	62.85	0.7043
Experiment5	71.82	67.70	68.23	0.8014

In the below, we have showed train accuracy comparison of different Machine Learning Model such as Decision Tree, Naive Bayes, Logistic Regression and Neural Network model using LSTM.

TABLE II
PERFORMANCE COMPARISON USING DIFFERENT MODELS

Input	Decision Tree	Naive Bayes	Logistic Re.	LSTM
Experiment	92	80	75	80.14

VI. CONCLUSION AND FUTURE DIRECTIONS

In conclusion, the experimental results show that our multi-step classifier based on Long Short-Term Memory (LSTM) model is effective in recognizing music genres. For 10-genre classification, the accuracy was 60% using a single LSTM[4]. But we achieved an accuracy of 80.00%, which was better than one of the four genre classification approach having an accuracy of 51.88%. There is no denying that since all of this

research has had to deal with the same issues in GTZAN, the results remain comparable. Hence, our future aim is to improve our accuracy by using more LSTM layers, adding dense layers.

REFERENCES

[1] N. Pelchat and C. M. Gelowitz, "Neural network music genre classification," in *2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE)*. IEEE, 2019, pp. 1–4.

[2] "Gatzan Dataset-Music Genre Classification," https://www.kaggle.com/andradaolteanu/gtzan-dataset-music-genre-classification.

[3] Y. M. Costa, L. S. Oliveira, A. L. Koericb, and F. Gouyon, "Music genre recognition using spectrograms," in *2011 18th International conference on systems, signals and image processing*. IEEE, 2011, pp. 1–4.

[4] M. Lopes, F. Gouyon, A. L. Koerich, and L. E. Oliveira, "Selection of training instances for music genre classification," in *2010 20th International Conference on Pattern Recognition*. IEEE, 2010, pp. 4569–4572.

[5] C. P. Tang, K. L. Chui, Y. K. Yu, Z. Zeng, and K. H. Wong, "Music genre classification using a hierarchical long short term memory (lstm) model," in *Third International Workshop on Pattern Recognition*, vol. 10828. International Society for Optics and Photonics, 2018, p. 108281B.