CS 464 – Introduction to Machine Learning

Project Final Report

Project Name: Music Genre Classifier

Group 20

Fahad Waseem Butt - 21801356

Roshaan Abbas Jaffery - 21901284

Maaz Ud Din - 21901258

Payam Sedighiani - 21801298

TABLE OF CONTENTS

1.0 INTRODUCTION

In this project, we set out to explore Machine Learning Classification methods. We proposed several methods, such as some standard Machine Learning methods and some Deep Learning methods. Considering our options, we ended up choosing to do three methods, the k-Nearest Neighbours, a Convolutional Neural Network and a Long Short-Term Memory learning method. Our data was .wav audio files, so the first step was to preprocess the data by taking the audio data and converting it to the Mel Spectrogram format. Then, for the k-Nearest Neighbours method, a Principal Component Analysis was conducted to reduce the dimensionality in the data to fit the method better. With these steps done, the actual methods were implemented and the results were gathered and analyzed.

2.0 PROBLEM DESCRIPTION

The main goal of this project was to classify instrumental audio data into the genres of the audio - a music genre classifier. The first step in accomplishing this was to conduct an exploratory data analysis of the data, wherein the audio data was judged on how the features in it could be used in the methods we wanted to implement. As the regular time-series data is not suitable to properly solve a classification problem, the data needed to be converted to another form where the features are expressed more distinctly. To achieve this, the time-series data was converted to the Mel Scale, where the features had much more variance to help in classification; more elaboration on this will come in a later section.

We chose to work with machine learning algorithms of varying levels of complexity, namely the k-Nearest Neighbours, Convolutional Neural Networks and Long Short-Term Memory methods. The challenges presented in implementing these methods primarily consisted of understanding the methods, understanding how the hyperparameters of the methods work and how to evaluate the methods and compare them to each other. The primary metric used to evaluate model performance was the model's accuracy, with some other metrics like Precision and Recall also being considered.

Another constraint on working on this project was the time, which limited us to work with three machine learning algorithms. Depending on how the timeline for working on this project progresses, we may explore further on how the data may be processed and how the models can be optimized to improve the results.

3.0 METHODS

3.1 Dataset Description

The GTZAN Genre Dataset [1] was used to work on this project. This dataset can be found on Kaggle, and was originally used in 2001 in the paper "Automatic Musical Genre Classification Of Audio Signals" [2]. It has a collection of 30 seconds instrumental songs of 10 genres, making it a dataset of 10 classes, with each class having 100 song tracks. In total, this dataset has 1000 audio '.wav' data samples each 30 seconds long. The genres included are "blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae, and rock" [1]. For each audio file, there is also an image version giving a visual representation of the song files.

Furthermore, there are 2 CSV files (for 30 seconds audio and for a 3 seconds audio for each) containing some extracted features.

Additionally, when the data was being preprocessed, it was found that there was one corrupt file in the data. Namely, the file “jazz.00054.wav” was corrupt, it was found to not be usable through error logs, and when manually playing this specific file from the device, the audio was not playing and instead giving an error. To put this into context, none of the other audio files gave such errors. To resolve this, the file “jazz.00099.wav” was duplicated and renamed “jazz.00054.wav” to replace the original. This way, the number of data files in each class remained as 100, with a duplicate file in the Jazz genre, but this was not expected to, and did not, impact the training and results of the machine learning methods.

3.2 Preprocessing

For this project, we have been working with audio data, that is in the time-series data form. This can be visualised as follows when the data is directly plotted, as can be seen in Figure 1.

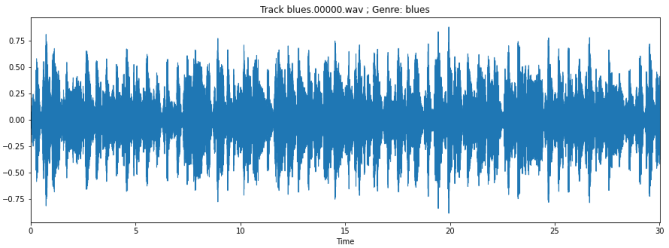

Figure 1: Blues 00 Time-Series Data Visualised in a Plot

Figure 1 shows an example for the time-series plot for the raw audio data. This data cannot directly be used to train the models, and so a Short-Time Fourier Transform (STFT) is taken to attempt to convert it to a more usable form, and a plot of this can be seen in Figure 2.

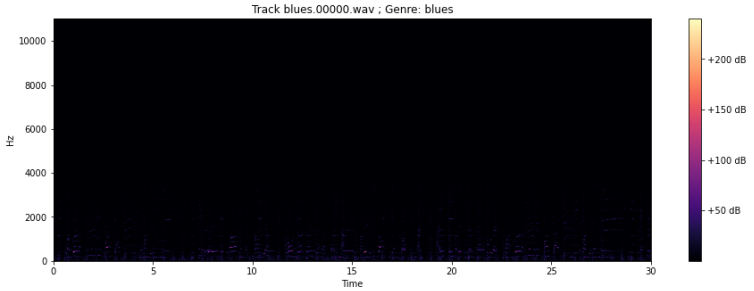

Figure 2: Blues 00 Data Visualised in a Plot after a Short-Time Fourier Transform

In Figure 2, it can be observed that there is very little data that can be used for training. The plot itself has very little observable data, with most of it being blank. Hence an alternative method to extract the data was needed to work on this problem. We might try to introduce nonlinearity in the data, by methods such as taking squares or logs. to increase the amount of data and variance in it, but with such data, it was determined to not be a feasible option and so this option was discarded.

Hence the method chosen to expand on usable features was to convert the data to the Mel scale. The Mel scale was proposed in 1937 by Stevens, Volkmann and Newmann, and

converts audio data to a form more similar to how the human ear perceives music. The scale is composed of pitches determined by listeners to have an equal distance from each other. In the Mel scale and regular frequency measurement, the reference point is “equating a 1000 Hz tone, 40 dB above the listener's threshold, with a pitch of 1000 mels”. The scales coincide when below 500 Hz, but the Mel and Hertz scales are judged by listeners to have larger intervals on larger intervals so as to get the same pitch increases [3].

To get an image Mel Spectrogram from the music data, the librosa package [4] was used for analyzing the audio. To specifically get the frequency spectrum on the Mel scale, “librosa.feature.melspectrogram”[5] was used. The hyperparameters modified were as follows: the time-series audio data, sampling rate (set at 32,000 Hz so there is lower loss of data), the length of the Fast Fourier Transform (FFT) window (traditionally powers of 2 are used, so we chose 2^{11} = 2048; using FFT windows simulates a STFT), the hop length (the number of samples between successive frames, and it should be less than the FFT window length, so it was chosen as 1024), and number of Mel bands to generate (which was chosen as 128 Mel bands as per tradition). The following figures show Mel Spectrogram results for each class.

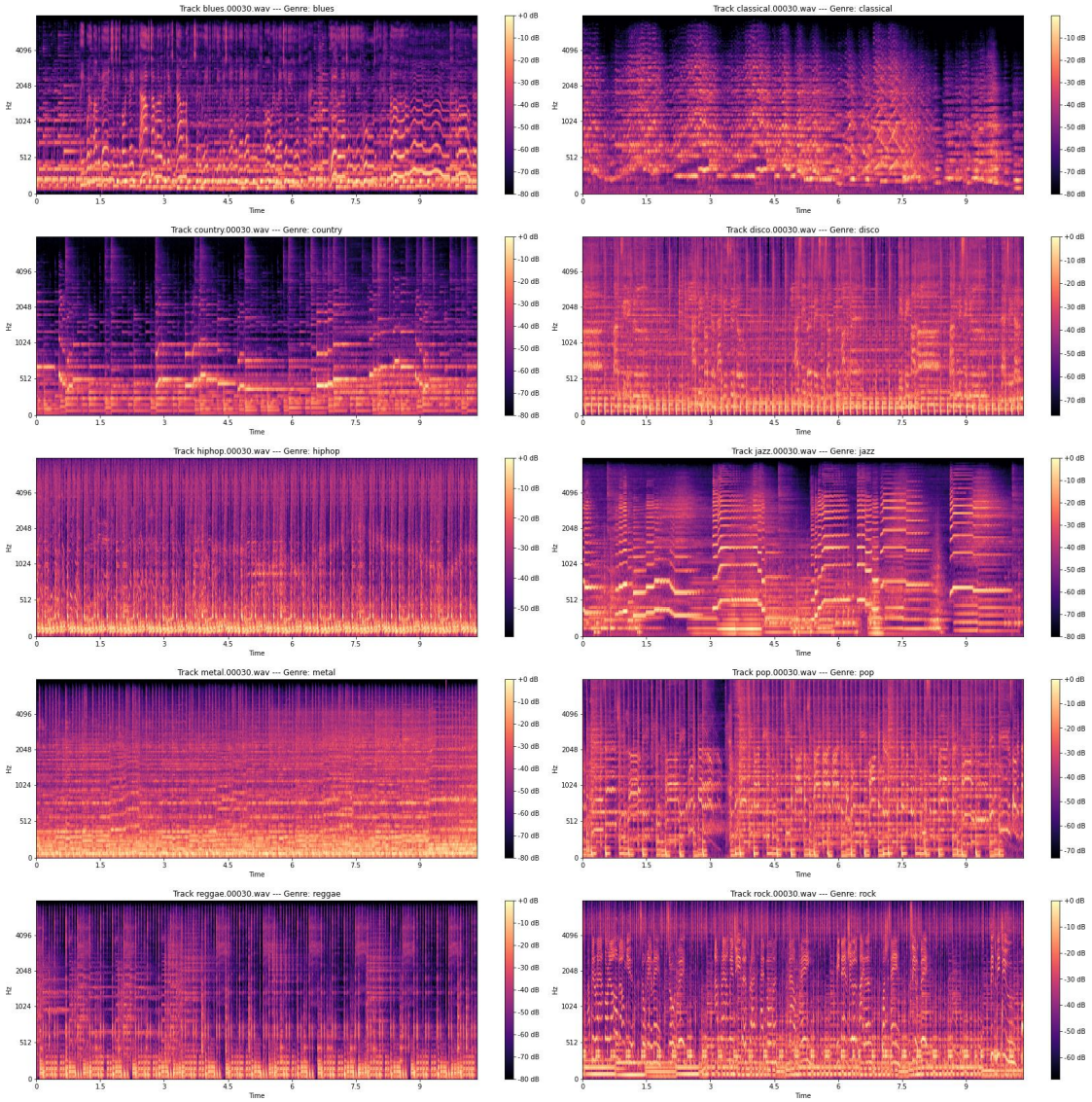

Figure 3: Mel Spectrograms the 30th Data File Visualised in a Plot for Each Genre

Figure 3 shows the plots for an audio sample from each genre in the dataset, and it can now be observed that the spectrogram shows much clearer data with much more variance. Even looking at the spectrograms quickly the plots display differences visible to a layman observer. So, for each data point, 81920 features were found.

Hence, the data exploration step has been completed and it can be used much more easily in training our machine learning models. After this, the data was split into training and test splits, with 80% being used for training and 20% being used in testing. Additionally, for the Neural Network Models, a separate preprocessing is conducted for each, with slight differences, wherein the same procedure is applied, just with taking 3 second clips of each audio file, making for 10,000 data points in total and 1000 3 second audio tracks in total. This increases the number of data points by a factor of 10 multiplicatively and helps capture further variance in the data.

3.3 Principal Component Analysis

Principal Component Analysis (PCA) is a technique that is used to reduce the dimensions of large data sets, by the transformation of a large set of variables into smaller sets of variables while still containing most of the information from the larger set. Usually, accuracy is compromised while reducing the dimensions (variables) of a data set, but this increases the simplicity of the set, as reduced variables allow for easier and faster analysis for the ML algorithms. The variables which do not affect the outcome largely (the extraneous variables) are excluded during PCA. So the goal during PCA is to preserve as much information as possible, while reducing the variables in a set. In short, the steps for PCA are: standardisation; covariance matrix computation; computing eigenvalues and eigenvectors; identifying the principal components[6]. The standardisation is done using the following formula:

$$z = \frac{value - mean}{standard\ deviation}$$

Then the covariance matrices are constructed to analyse how the variables are varying from the means w.r.t each other. The matrix is a $n \times n$ symmetric matrix. An example of a 3×3 covariance matrix is as follows:

$$\begin{bmatrix} Cov(x,x) & Cov(x,y) & Cov(x,z) \\ Cov(y,x) & Cov(y,y) & Cov(y,z) \\ Cov(z,x) & Cov(z,y) & Cov(z,z) \end{bmatrix}$$

Next, the eigenvalues and eigenvectors are calculated to create the linear combinations of the initial variables, which are our principal components. This is done in such a way that the emerging variables contain most of the information from the initial variables, but are uncorrelated. PCA tries to gather most of the information in the initial components, hence reducing the dimensions of the set. The variable with the largest possible variance in the dataset is made the first principal component and so on. All the principal components are uncorrelated with each other [7].

In our case, we have 1000 audio files, with 81920 features in each audio data file. The following plot shows the explained variance for the features contained in N PCs. About 75%

variance is explained by the first 190 PCs, and around 500 PCs are needed to improve it to 100% of variance explained. After this point, increasing the number of PCs gives diminishing returns in variance explained. This can also be seen in Figure 4. Hence a PCA being applied to the data would help make the classification more efficient.

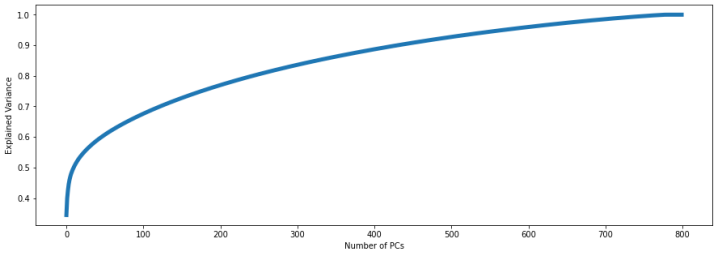

Figure 4: PVE varying with the number of PCs shown on a plot

3.4 k-Nearest Neighbours

The k-Nearest Neighbours method (kNN) is a basic ML method, but has complexities that can cause highly varied results depending on the type and amount of data used.

The method to run the kNN method is as follows: choose a k value, which is the number of nearest neighbours that will vote on what to classify a datapoint as; find the distance, based on some distance metric, of each test point with all points in the training data; choose the k number of least distances and take a majority vote amongst them; classify the datapoint being tested to the class with the most votes [8]. The distance metric used in the project was the Euclidean Distance which is calculated as follows.

$$d(x, y) = \sqrt{(y - x)^2}$$

The kNN algorithm works differently for each unique value of k, however there is no correct method to choose a value of k. A k value that is too high can cause a high bias and low variance. A k value that is too low can cause low bias and high variance [9]. This can cause problems in larger datasets, as when there are a large number of features, a high value of k is required, but the kNN method performs worse when k gets too high. Hence, due to the curse of dimensionality, the k-Nearest Neighbours algorithm is limited on how well it can perform after the number of features in the data become too much. In this case, there is also the issue of time efficiency and memory, because it needs to perform multiple calculations for each data point, and this becomes a slower process on very large datasets. Therefore, kNN on large datasets needs to be preceded by a dimensionality reduction method such as the Principal Component Analysis.

3.5 Convolutional Neural Network

When discussing Deep Learning methods, Convolutional Neural Networks are popular because of how well they can learn and how they can effectively be used to train for different types of classifications [10]. For this purpose of this project, which is to classify audio which has been converted into a Mel Spectrogram format, we used a CNN architecture for training the model. Since the audio features now are similar to image type data, the CNN is an appropriate neural network model which allows for extraction of higher representations for the image content [11].

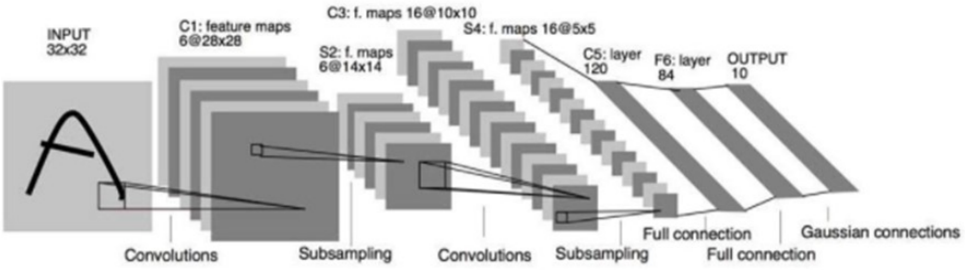

Figure 5: CNN Architecture [11]

As can be seen in Figure 5, a Convolutional Neural Network extracts features from images through a combination of Subsampling and Convolution. It produces feature maps, which are the convolved features, and based on the filter value, the feature maps will change [11]. More filter values imply more features being extracted, and hence, a more accurate estimation for the labels.

In our case, as we have 1000 audio files already preprocessed into an Mel spectrogram form, we used those mel spectrograms as our main data input into the CNN model to extracts the features from those images as genres being their labels and which helped in training our model and then give predictions according to the validation set made in the aforementioned preprocessing methods. In our CNN model, we used three convoluted layers for our model which helps in developing a feature map via filter and activation process which indicates the locations and strengths of the detected feature in that input, in our case the Mel spectrograms of the audio files [12].

3.6 Long Short-Term Memory Network

Long Short-Term Memory (LSTM) Networks are another form of neural networks, more specifically a specific kind of Recurrent Neural Network (RNN). RNNs are able to learn long-term dependencies in the data, and work particularly well when dealing with problems that involve prediction a sequence [13], this means that it can work well on audio and image data, such that it can learn better from such data. With LSTMs being, essentially, better RNNs, this model is a good fit for this project. LSTM holds memory in a memory cell called the "cell state" [13]. The overall model of the LSTM is visualized in Figure 6.

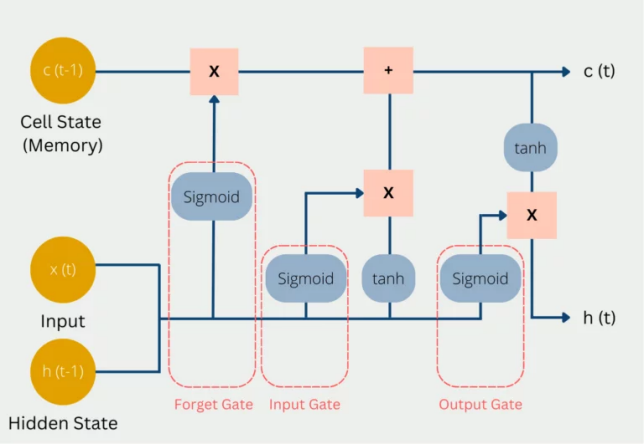

Figure 6: Logic in the LSTM Architecture [14]

The cell state stores the short-term memory, but there is also a hidden state, as shown in Figure 6. Values pass through gates that regulate the data before the states. The gates are: the Forget Gate; the Input Gate; the Output Gate [14]. The Forget Gate decides which of the previous and current information is saved or discarded, done so by passing the data through a sigmoid to get values between 0 and 1, and then multiplying with the cell state to drop out unnecessary information. The Input gate determines which of the input is valuable for the task at hand and which is not, and does this by multiplying the current input by the hidden state and previous weight matrix, so as to add the important information to the cell state to update it [14]. The Output Gate calculates the output of the LSTM in the hidden state by applying a sigmoid and then multiplying with the tanh function after the activation [14].

In this project, we made use of the Bidirectional LSTM model, which has connections at every point in the Bidirectional layer, allowing it to have information for each part of a sequence in both forwards direction and the backwards direction [13]. For this key reason, Bidirectional LSTMs are very good at learning data more precisely, allowing for better results when making predictions, and so, making them a good fit for this project.

4.0 RESULTS

4.1 k-Nearest Neighbours Results

The k-Nearest Neighbours method was run for different values of k checking from k=5 to k=10 neighbours to determine the best value for k. The kNN method used data with reduced dimensionality from PCA, with 18 Principal Components used. The results in Table 1, Table 2 and Figure 7 were found for this implementation.

k (number of neighbours)	Accuracy	Accuracy / %
5	0.441 (+/-0.055)	44.1
6	0.447 (+/-0.053)	44.7
7	0.451 (+/-0.053)	45.1
8	0.453 (+/-0.078)	45.3
9	0.464 (+/-0.046)	46.4
10	0.449 (+/-0.061)	44.9

Table 1: k-Nearest Neighbours Accuracy Results

Class (Genre)	Recall	Precision	F1-Score
0	0.33	0.44	0.38
1	0.67	0.44	0.53
2	0.43	0.39	0.41
3	0.31	0.33	0.32
4	0.45	0.82	0.58
5	0.14	0.25	0.18
6	0.70	0.78	0.74
7	0.74	0.56	0.64
8	0.62	0.23	0.33
9	0.20	0.11	0.14
Macro Average	0.46	0.44	0.42
Weighted Average	0.47	0.43	0.43
Average Accuracy	0.43		

Table 2: k-Nearest Neighbours Detailed Classification Report

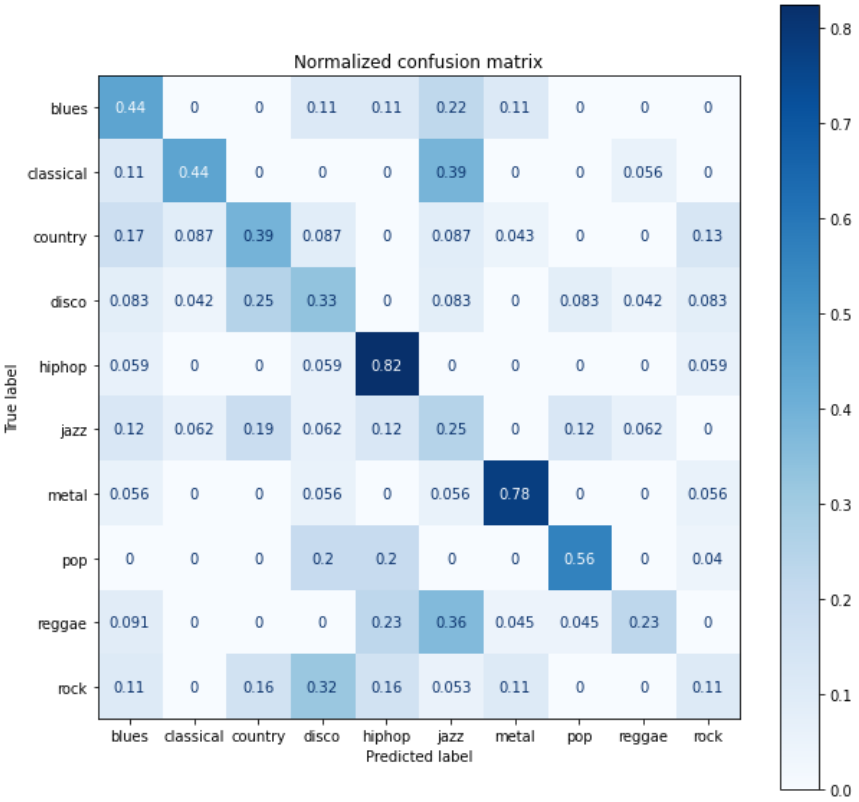

Figure 7: k-Nearest Neighbours Normalized Confusion Matrix

4.2 Convolutional Neural Network Results

The Convolutional Neural Network model was run for 50 Epochs and had three Convolutional Layers (with Batch Normalization and Max Pooling), with the later two Convolutional Layers having Dropout. The activation function used was RELU. This was followed by a Flatten Layer, a Dense Layer, a Dropout Layer and another Dense Layer with a Softmax activation. The Adam optimizer was used with a batch size of 128. The loss function was Sparse Categorical Cross Entropy Loss, Figure 8 and Figure 9 show the results of the CNN method.

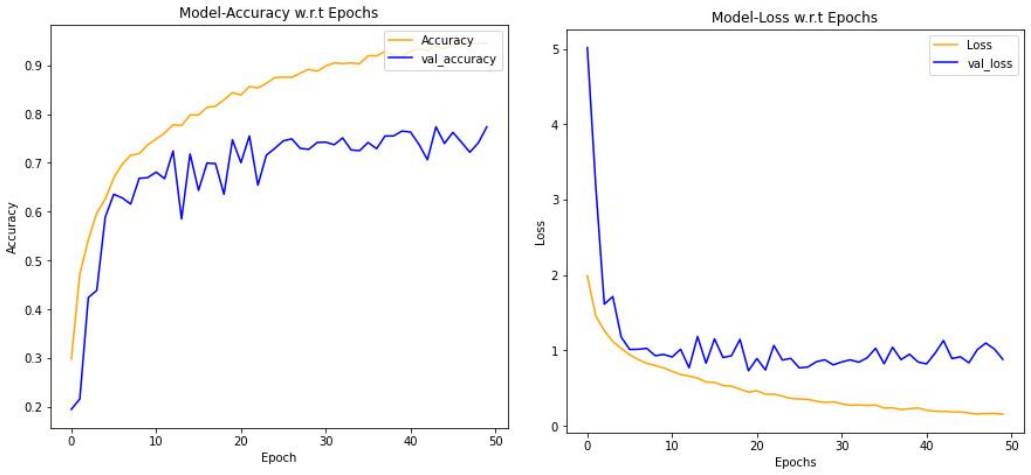

Figure 8: Convolutional Neural Network Accuracy and Loss Curves

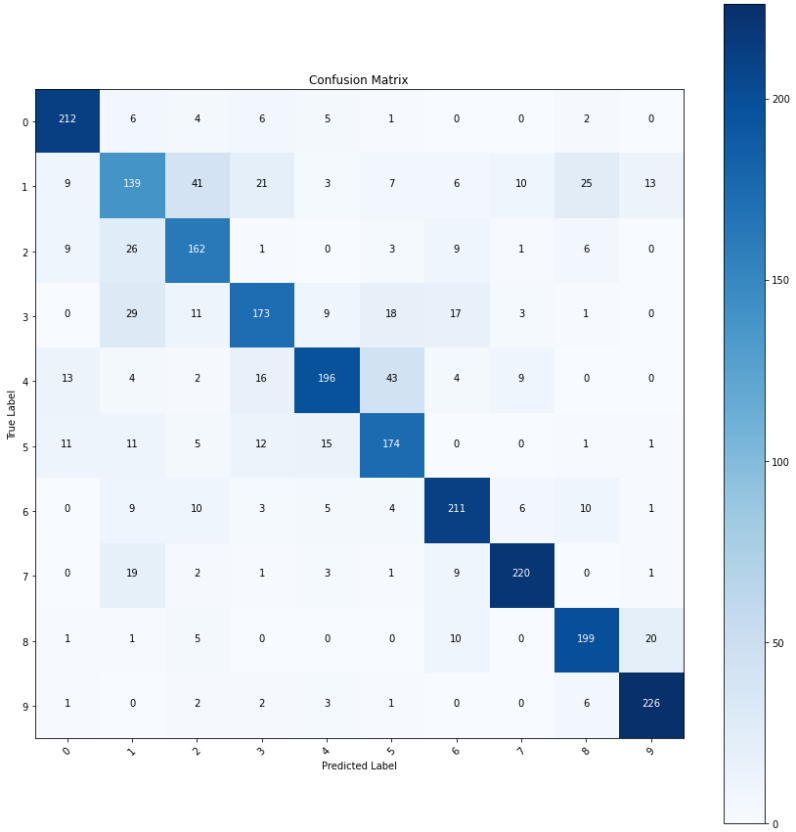

Figure 9: Convolutional Neural Network Confusion Matrix

4.3 Long Short-Term Memory Results

The Long Short-Term Memory model was run for 100 Epochs and had a Bidirectional layer is for the LSTM with 512 Units, Activation='tanh', recurrent_activation='sigmoid', using both dropout and recurrent dropout, returning sequences. A Flatten layer is added and a Dense layer is added with a “softmax” activationThe Adam optimizer was used with a batch size of 128. The loss function was Categorical Cross Entropy Loss, Figure 10 and Figure 11 show the results of the LSTM method.

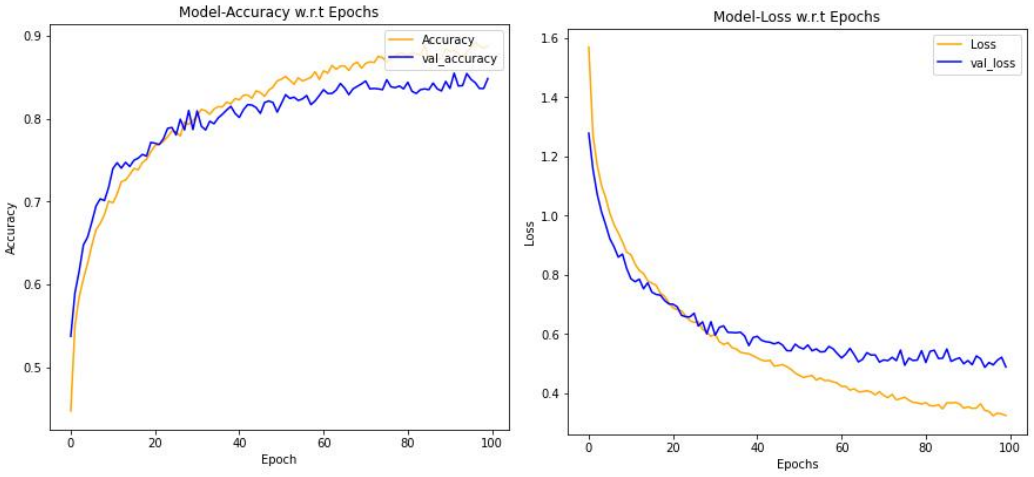

Figure 10: Long Short-Term Memory Network Accuracy and Loss Curves

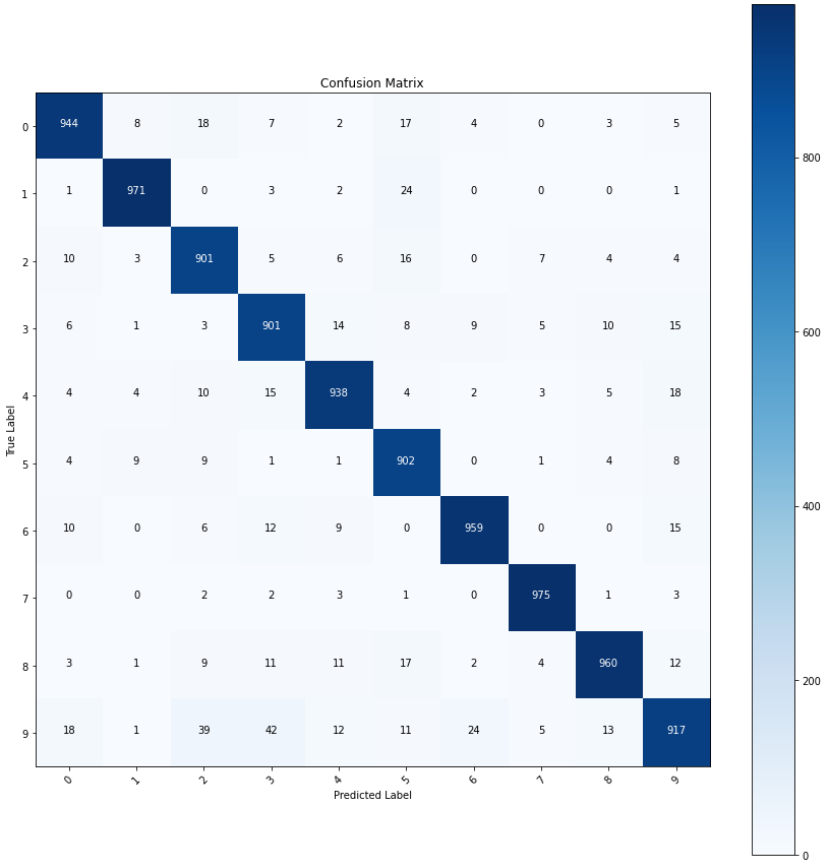

Figure 11: Long Short-Term Memory Network Confusion Matrix

5.0 DISCUSSION

5.1 k-Nearest Neighbours Discussion

It can be seen that the best parameter k for k-Nearest Neighbours to be used is k = 9 as it gives an accuracy of 46.4%. The overall accuracy of the model is 43%, which is expected for a kNN classifier on larger datasets.

This is because, when the dataset increases, kNN falls to the curse of dimensionality. With a larger dataset, there is a need for a higher value of k, but higher values of k there is high bias and low variance, leading to the model fitting poorly. To mitigate this, Principal Component Analysis was used to reduce the dimensionality of the data, with it being already found that about 200 Principal Components are required to capture 80% of the variance in the data. But using such a value of k gave even lower accuracy results, meaning that even after reducing the 81920 to 200, there was still too much dimensionality in the data.

So the best result we could obtain was the accuracy of 46.4% with k = 9. The best classified genres were Pop, Metal and Hiphop, which seems correct because of the uniqueness of how they each sound.

5.2 Convolutional Neural Network Discussion

The Convolutional Neural Network method gives a final validation accuracy of 77.37% after convergence, which is much higher than the accuracy obtained from k-Nearest Neighbours. At the last epoch, the training loss was 0.1566, the training accuracy was 94.49%, whereas the validation loss was 0.8832. So the model itself is definitely learning.

The Adam optimizer was chosen for this part of the project. The loss function is chosen traditionally to be Sparse Categorical Cross Entropy loss, and that worked well for us.

But despite the model learning to classify the music genres to a reasonable extent, it can be seen that, while the training loss and accuracy converge to almost ideal values, the testing loss and accuracy reflect that the model is slightly overfitting. However, since the validation accuracy is above 75%, such a small difference in the metrics for training and validation does not imply a very strong level of overfitting, and that the model is trained properly. This result is acceptable, and actually rather good, as is justified in the confusion matrix where the concentration of True Positives in the diagonal is rather high.

5.3 Long Short-Term Memory Discussion

The Long Short-Term Memory Network method gives a final validation accuracy of 84.83% after convergence, which is higher than the accuracy obtained from k-Nearest Neighbours, as well as from the accuracy for the Convolutional Neural Network. At the last epoch, the training loss was 0.3239, the training accuracy was 88.73%, whereas the validation loss was 0.4880. So the model itself is definitely learning.

The Adam optimizer was chosen for this part of the project. The loss function is chosen traditionally to be Sparse Categorical Cross Entropy loss, and that worked well for us.

This time, the model is performing very well, in that the training and validation accuracies are very close to each other, hence it can be stated that there is almost no overfitting in the model. The overall training accuracy of the LSTM model may be a bit lower than the training accuracy of the CNN, but since the validation is not overfitting at all, it is actually a better result. Hence, this is a very good result, and is even justified by the confusion matrix, where the concentration of True Positives in the diagonal is even higher than in the CNN.

6.0 CONCLUSION

With the above discussion, our project has been completed and we accomplished all the goals we set out to do. We converted our audio data into a more usable form in Mel Spectrograms, and implemented the PCA, kNN, CNN and LSTM algorithms. We managed to improve the performance of the models by increasing the data samples extracted and optimized the methods we used. We were successful in getting good results for our neural network models and in proving that the kNN is not suitable for such a task. The project was a hands-on deep dive into Machine Learning methods, and we learnt a lot, and that goes especially so for how much we learnt about Neural Networks.

7.0 APPENDIX

Fahad Waseem Butt worked on understanding and then Preprocessing the audio data by converting it from a time-series to the Mel Scale. Aside from this Fahad Waseem also worked with the other group members in the other portions of the project, working on understanding and implementing the Principal Component Analysis, k-Nearest Neighbours, the Convolutional Neural Network and the Long Short-Term Memory methods, both to make them work and in exploring possibilities to optimize them.

Roshaan Abbas Jaffery worked primarily on the implementation of the Convolutional Neural Network model, working to understand and implement it effectively, and after the progress presentation, worked on improving the model even more. Roshaan worked on this with Maaz Ud Din and Fahad Waseem. Roshaan also assisted Fahad Waseem in some parts of the Preprocessing of the dataset, as well as to work in increasing the number of samples that were being extracted from the data.

Maaz Ud Din worked with Roshaan Abbas Jaffery on the Convolutional Neural Networks implementation and on how the results needed to be improved. Maaz also worked on the Principal Component Analysis section, and worked with Fahad Waseem in the implementation. He also worked with Fahad Waseem on improving the Long Short-Term Memory Network that was made.

Payam Sedighiani worked primarily on the k-Nearest Neighbours method, in understanding how it works and also in how to implement it properly. Payam worked on this with help from Fahad Waseem. He also provided input when Roshaan and Fahad were considering how to work on the Preprocessing, and helped in working on making the Preprocessing for both the Convolutional Neural Network and Long Short-Term Memory.

All of us worked collectively to work on the report, and divided the sections amongst us. We finished it collectively, as a unit.

8.0 REFERENCES

[1] "GTZAN Dataset - Music Genre Classification," Kaggle, 2019. [Online]. Available: https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification

[2] G. Tzanetakis, G. Essl, P. Cook, "Automatic Musical Genre Classification Of Audio Signals," The International Society for Music Information Retrieval (ISMIR), 2001. [Online]. Available: https://ismir2001.ismir.net/pdf/tzanetakis.pdf

[3] "MEL," Simon Fraser University. [Online]. Available: https://www.sfu.ca/sonic-studio-webdav/handbook/Mel.html

[4] McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., & Nieto, O., "librosa: Audio and music signal analysis in python", 2015, In Proceedings of the 14th python in science conference (Vol. 8).

[5] "librosa.feature.melspectrogram," Librosa.org. [Online]. Available: http://librosa.org/doc/main/generated/librosa.feature.melspectrogram.html#librosa.feature.melspectrogram

[6] "What Is Principal Component Analysis (PCA) and How It Is Used?" Sartorius, 2020. [Online]. Available: https://www.sartorius.com/en/knowledge/science-snippets/what-is-principal-component-analysis-pca-and-how-it-is-used-507186

[7] Z. Jaadi, "A Step-by-Step Explanation of Principal Component Analysis (PCA)," Builtin, 2022. [Online]. Available: https://builtin.com/data-science/step-step-explanation-principal-component-analysis

[8] D. L. Yse, "K-Nearest Neighbor (KNN) Explained," Pinecone. [Online]. Available: https://www.pinecone.io/learn/k-nearest-neighbor/

[9] T. C. Lin, "Day 3 — K-Nearest Neighbors and Bias–Variance Tradeoff," Medium, 2018. [Online]. Available: https://medium.com/30-days-of-machine-learning/day-3-k-nearest-neighbors-and-bias-variance-tradeoff-75f84d515bdb

[10] S. Saha, "A Comprehensive Guide to Convolutional Neural Networks - the ELI5 way," Towards Data Science, 2018. [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[11] V. Tatan, "Understanding CNN (Convolutional Neural Network)," Towards Data Science, 2019. [Online]. Available: https://towardsdatascience.com/understanding-cnn-convolutional-neural-network-69fd626ee7d4

[12] J. Brownlee, "How Do Convolutional Layers Work in Deep Learning Neural Networks?," Machine Learning Mastery, 2019. [Online]. Available: https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/

[13] “What is LSTM - Introduction to Long Short Term Memory,” Intellipaat, 2022. [Online]. Available: https://intellipaat.com/blog/what-is-lstm/

[14] “Long Short-Term Memory Networks (LSTM)- simply explained!,” Data Basecamp, 2022. [Online]. Available: https://databasecamp.de/en/ml/lstms