COMP47650 – Deep Learning Project

Adam Morrissey
University College Dublin
19359553
adam.morrissey1@ucdconnect.ie

Abstract

This project addresses the problem of automatic music genre classification using deep learning techniques. By employing various models including Multi-Layer Perceptron (MLP), Convolutional Neural Network (CNN), and Recurrent Neural Network (RNN) within the TensorFlow framework [1], I classified the GTZAN music dataset [2] into 10 different genres. The experimental results demonstrate notable distinctions in performance across these models. The CNN and RNN models, in particular, showcased higher accuracy and reliability in genre classification, achieving accuracy scores of up to 72% and 79% respectively. These results highlight the effectiveness of deep learning techniques in the music classification domain, paving the way for further exploration into more sophisticated audio processing methods.

1 Introduction

Music has always been fundamental to human culture, conveying emotions, memories, and ideas for centuries. With the digitization of music, efficiently managing and categorizing vast libraries has become increasingly challenging. Automated systems that analyze, categorize, and recommend music are essential tools for both music streaming industries and consumers navigating the vast musical landscape.

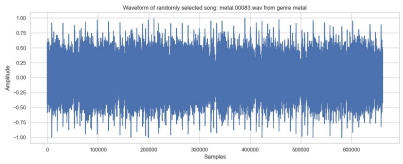
While music classification has traditionally focused on genres, recent advancements have expanded to include mood and emotion, improving music recommendations. This study leverages the GTZAN dataset, created by George Tzanetakis and Perry Cook in 2001, which comprises 1,000 tracks across 10 genres. We employ deep learning models—Multi-Layer Perceptron (MLP), Convolutional Neural Network (CNN), and Recurrent Neural Network (RNN)—to classify these tracks, with a focus on Mel Frequency Cepstral Coefficients (MFCCs), a technique critical for distinguishing musical timbre.

The paper first reviews related work in music genre classification, followed by a detailed discussion of the experimental setup and deep learning models used. Later sections explore potential model deployment architectures and conclude with a summary of findings and future research directions.

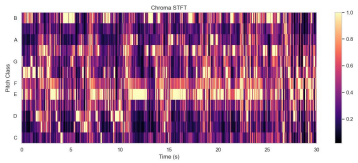
2 Dataset

The GTZAN Music Genre Classification dataset [1] contains music samples from 10 genres, each with a uniform duration of 30 seconds. Each 30-second track is subdivided into ten 3-second segments to increase the dataset's volume, providing more samples for training.

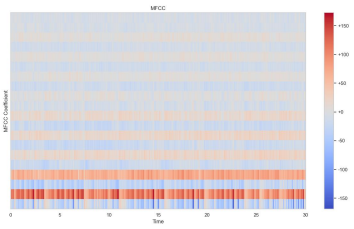
[1] https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification/data



(a) Waveform of a Song



(b) Chroma STFT



(c) MFCC

Figure 1: Example plots of different audio features: (a) Waveform of a song, (b) Chroma STFT, (c) MFCC

2.1 Feature Extraction

The audio features were extracted using the `librosa` package [3],converting the raw audio into a spectrogram to capture frequency content over time. Various features were investigated, including Chroma STFT, RMS Mean, Spectral Centroid Mean, Spectral Bandwidth Mean, Rolloff Mean, Zero Crossing Rate Mean, Harmony Mean, and Mel Frequency Cepstral Coefficients (MFCC). The mean and variance were calculated for most of these features, and after extensive study, MFCC proved the best focus for deep learning. [4] [5]

2.1.1 Visualizations of Audio Features

To better understand the distribution and structure of features in the GTZAN dataset, we present multiple visualizations:

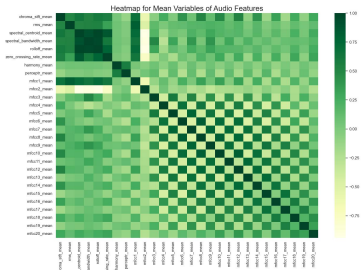
- **Heatmap Analysis**: The heatmap visualizes the relationships between the mean values of various audio features. This matrix helps identify the strengths and directions of these feature relationships.
- **Box Plot for BPM Distribution**: The box plot shows the distribution of Beats Per Minute (BPM) across music genres, indicating the central tendency, spread, and outliers for each genre's BPM distribution.
- **PCA on Genres**: The Principal Component Analysis (PCA) plot displays genre groupings based on their audio features, reducing the high-dimensional feature space into two principal components to make genre clusters visually distinguishable.

2.2 Machine Learning Classification

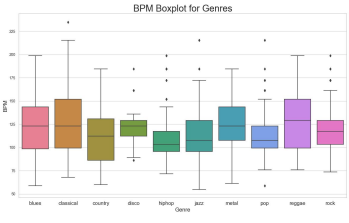
Pre-trained models were created using scikit-learn [6] and XGBoost [7] libraries for machine learning classification. The accuracies for different models are presented in the following table:

3 Related Work

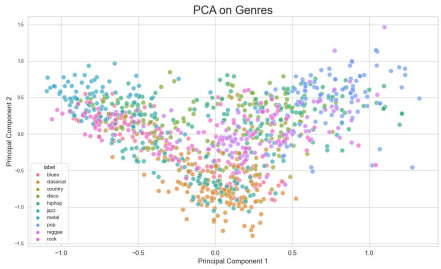
Recent advancements in music classification research have increasingly emphasized understanding the emotional content of audio signals. This evolution has involved using spectral and harmonic features alongside neural network models. Bhat et al. introduced a technique that used these features to infer the mood of music, focusing on rhythm, harmony, and spectral characteristics based on Thayer's model, which suggests a relationship between musical features and human perception [8].



(a) Heatmap for Mean Variables of Audio Features



(b) Box plot for BPM Distribution among Genres



(c) PCA Visualization of Genre Groupings

Figure 2: Visualizations of Audio Features: (a) Heatmap, (b) Boxplot for BPM Distribution, and (c) PCA of Genre Groupings

Model	**Accuracy**
Naive Bayes	0.51952
Stochastic Gradient Descent	0.65532
K-Nearest Neighbors	0.80581
Decision Tree	0.64665
Random Forest	0.81415
Support Vector Machine	0.75409
Logistic Regression	0.69570
XGBoost	0.90123
XGBoost Random Forest	0.74741

Table 1: Accuracies of different machine learning models

Similarly, Kim et al. designed a probability-based model for mood-based music recommendations, implementing it to test user satisfaction [9].

Patel et al. used digital signal processing techniques to differentiate musical notes by analyzing pitch and loudness, underscoring the importance of feature extraction in music analysis [10]. In contrast, Tzanetakis and Cook pioneered genre classification by leveraging features like texture, rhythm, and pitch, using machine learning models to analyze real-world audio collections [2].

More recent approaches have leveraged convolutional neural networks (CNNs) and support vector machines (SVMs) to improve classification accuracy. Choi et al. demonstrated automatic tagging using deep CNNs [11], while Han et al. combined CNNs and SVMs for genre classification [12]. Li et al. incorporated attention mechanisms into CNNs to improve genre classification performance [13]. These studies illustrate that combining traditional machine learning with modern deep learning techniques can significantly enhance music classification accuracy.

Our work builds on this progress by leveraging the GTZAN dataset to train MLP, CNN, and RNN models for genre classification. By focusing on extracting robust audio features like Mel Frequency Cepstral Coefficients (MFCCs) and optimizing the models, this project aims to further refine the accuracy of music classification systems.

4 Experimental Setup

The GTZAN dataset, collected in 2001, contains 10 genres with 100 samples each, making it a well-balanced dataset for genre classification. Each audio track is 30 seconds long and has been sampled at a rate of 22,050 Hz, providing high-quality data for analysis. The dataset does not have significant imbalance issues, allowing straightforward model training without requiring additional balancing techniques.

4.1 Preprocessing

Preprocessing is essential to prepare the GTZAN dataset for deep learning models. The dataset is augmented by splitting each 30-second audio file into ten 3-second segments, yielding 10,000 data points and saved as a JSON file. This augmentation increases the training data size, enhancing model robustness. The `librosa` library extracts Mel Frequency Cepstral Coefficients (MFCCs) to capture the spectral envelope of audio, critical for genre classification. Each segment contains 13 MFCCs that represent each song track. Problematic files, such as `jazz.00054.wav`, are skipped to avoid errors. Data is split into training and testing sets with a 70:30 ratio. The training data is further split, creating a validation set from 20% of the training data. The data is shuffled using a fixed random seed for consistent splits. Depending on the model (CNN or RNN), a channel dimension is added to the input data. The processed dataset is then ready for training and evaluation.

4.2 Proposed Algorithms

Three deep learning models were designed to classify music genres: MLP, CNN, and RNN. The MLP model transforms the input data into a one-dimensional array and passes it through three dense layers to learn features. The CNN model uses convolutional layers to extract spatial features, and the RNN (LSTM) model captures long-term dependencies in sequential data. All models were trained using GPU resources to expedite the training process and efficiently handle the large dataset.

4.2.1 Baseline Approach

The baseline models were standard machine learning models such as Naive Bayes, K-Nearest Neighbors, and Decision Trees. These models served as a performance reference for comparison against the deep learning models.

4.3 Multi Layer Perceptron

The Multi-Layer Perceptron (MLP) model [14] was the first architecture designed in this project to classify music genres using the GTZAN dataset. It consists of an input layer that reshapes the multi-dimensional input data, $\mathbf{X}$, into a one-dimensional array, $\mathbf{x}$, making it suitable for the subsequent dense layers. The model has three hidden dense layers with ReLU activation functions, containing 512, 256, and 64 units, respectively. The ReLU activation function, defined as $f(x) = \max(0, x)$, enables the hidden layers to effectively learn features from the data. Mathematically, each hidden layer is represented as $\mathbf{h}_i = f(\mathbf{W}_i\mathbf{h}_{i-1} + \mathbf{b}_i)$, where $\mathbf{W}_i$ and $\mathbf{b}_i$ denote the weight matrix and bias vector of the i^{th} layer, respectively, and $\mathbf{h}_0$ is the input data. The final output layer uses a softmax activation function to generate a probability distribution for classification, with units matching the number of classes. The softmax function for the input vector $\mathbf{z}$ is defined as $\text{softmax}(z_j) = \frac{e^{z_j}}{\sum k=1^C e^{z_k}}$, where C is the number of classes, and z_j is the j^{th} element of $\mathbf{z}$.[15][16]

4.3.1 Fixing Overfitting

To address overfitting, the MLP model was enhanced with L2 regularization and dropout layers:

- **L2 Regularization:** This technique adds a penalty to the loss function that is proportional to the square of the weights:

$$\mathcal{L}_{\text{L2}} = \mathcal{L}_{\text{original}} + \lambda \sum_{i=1}^{n} w_i^2$$

  where λ is the regularization factor. This discourages large weights and prevents overfitting by simplifying the model. [17] [18]

- **Dropout:** This method randomly deactivates a fraction of neurons during training. With a dropout rate of p, $N \times p$ neurons are deactivated in each forward pass. This encourages the model to learn robust features and reduces overfitting.[19]

Implementing L2 regularization and dropout improves the model's generalization and reduces overfitting, enabling it to better perform on unseen data.

4.4 Convolutional Neural Network

The CNN model was designed to classify music genres using the GTZAN dataset. CNNs have played a significant role in advancing deep learning, as discussed in Schmidhuber's overview of deep learning [20]. The architecture comprises three convolutional layers, each followed by max-pooling and batch normalization. Each convolutional layer uses 32 filters ($F = 32$), which convolve over the input using kernel sizes of 3×3 and 2×2 to extract spatial features, applying the ReLU activation function to introduce non-linearity [**?**]. The max-pooling layers reduce the spatial dimensions and improve computational efficiency.

After the convolutional layers, the model flattens the feature maps into a one-dimensional vector and feeds it into a dense layer with 64 units and ReLU activation to further learn high-level features [21]. A dropout layer follows, with a 30% rate, randomly deactivating neurons to prevent overfitting. The output layer contains 10 neurons, one for each music genre, and applies a softmax activation function to produce a probability distribution for classification.

4.5 Recurrent Neural Network (LSTM)

The last model built was a Recurrent Neural Network (RNN) model, specifically utilizing Long Short-Term Memory (LSTM) layers, which are designed for handling sequential data like audio by capturing long-term dependencies. The concept of RNNs has been discussed extensively by Rumelhart et al. [22], Jordan [23], and Elman [24]. Additionally, practical guidance on working with RNNs can be found in the TensorFlow documentation [25].

The model architecture starts with an LSTM layer that processes sequential data and returns a sequence of outputs. The subsequent LSTM layer refines the sequence and returns a single output representing the final state. A fully connected dense layer with 64 units and ReLU activation further processes the data, and a dropout layer with a 30% dropout rate helps reduce overfitting. The final layer is a dense output layer with units corresponding to the number of classes, applying a softmax activation to predict the probabilities of each music genre.

4.6 Compilation and Training

The models were compiled and trained using the Keras library [26], a high-level neural networks API, running on top of TensorFlow. This allowed for an efficient setup of model architecture, the convenience of stock layers, and straightforward model compilation and training processes. Each model was configured with the Adam optimizer for effective learning rate adjustments and used `sparse_categorical_crossentropy` as the loss function for multi-class classification:

$$w = w - \eta \frac{m}{\sqrt{v} + \epsilon}$$

$$\mathcal{L}(y, \hat{y}) = -\sum_{i=1}^{N} y_i \log(\hat{y}_i)$$

The performance of the models was evaluated based on their accuracy, comparing the predicted outputs with the true labels in the validation dataset. This approach underscores the models' ability to generalize and perform under varied data conditions.

4.6.1 Hyperparameter Tuning

Hyperparameters for each model were tuned systematically to find the optimal configuration. This involved:

- Adjusting the learning rate to balance between speed and convergence.
- Experimenting with batch sizes to optimize training efficiency.
- Testing different network depths and layer sizes for each model.

The specific hyperparameters used are detailed in the appendix (see Table 2).

5 Results

The models were evaluated using accuracy, F1-score, precision, and recall metrics for comprehensive genre classification assessment.

Evaluation was facilitated by the `eval.py` script, which loaded the latest model checkpoint and computed metrics like accuracy, precision, recall, and F1-score, alongside generating a confusion matrix.

5.1 Model Performance

- **MLP**: Achieved 59% accuracy, showing balanced precision and recall across most genres, excelling in "Metal" but facing challenges with "Country" and "Rock."
- **MLP with Regularization and Dropout**: Achieved 55% accuracy, with similar trends to MLP but slightly lower accuracy.
- **CNN**: Achieved 72% accuracy, performed well in "Classical," "Metal," and "Pop," but struggled with "Country" and "Rock."
- **RNN**: Achieved 79% accuracy, excelling in "Classical," "Metal," and "Pop" with high precision and recall, but facing challenges in "Country" and "Rock."

For detailed insights into the training and validation curves of each model generated from the code, please refer to Figure 3. Additionally, the confusion matrix and classification report for each model can be found in Figure 4 and Table 3.

Comparing baseline machine learning models to deep learning counterparts, accuracies ranged from 51.95% to 90.12%, showcasing differences in feature extraction and pattern recognition approaches. Traditional models rely on handcrafted features and statistical techniques, while deep learning models, especially RNNs, autonomously learn hierarchical representations from raw data, resulting in superior performance.

In our results, we can see that MFCC can help in enhancing the efficiency and effectiveness of our automated music genre classification system. By encoding audio signals into a compact feature space, MFCC effectively reduces signal dimensionality, enhancing computational efficiency and simplifying the complexity of machine learning methods employed in audio processing tasks like genre classification. As well, it can capture important characteristics of audio signals which enables it to discern genre-specific elements across various pitches and timbres of music, contributing significantly to the accuracy and robustness of our classification system.

6 Conclusion and Future Work

In this work, we demonstrated the efficacy of deep learning models—MLP, CNN, and RNN—in classifying music genres using the GTZAN dataset. Our evaluation indicated the RNN model's superior performance in terms of classification accuracy, followed by the CNN and MLP models. The use of MFCCs as the primary feature extraction method contributed significantly to the accuracy of our genre classification system.

However, further advancements can be made:

- **Audio Length and Formats:** It could be possible to experiment with different audio lengths to see if shorter or longer segments influence the models' performance. Additionally, testing other audio formats beyond WAV could yield insights into feature extraction nuances across formats.
- **Improving Dataset Size and Diversity:** Future work should incorporate more extensive data augmentation techniques to improve the models' performance by diversifying and increasing the quantity of training data. Also, exploring other music datasets can provide

a broader representation of genres, especially those with nuanced sub-genres, which the GTZAN dataset lacks.

- **Hyperparameter Tuning:** Further hyperparameter tuning could help in determining the best combination for each model. Adjusting the learning rate, batch sizes, and network architectures systematically can improve accuracy.
- **Exploring Other Features:** The current models used MFCCs and spectrograms, but further exploration into patterns and lyrics could enhance the models' ability to capture more nuanced musical features, leading to improved accuracy.
- **Transfer Learning:** Utilizing pre-trained models like VGG or ResNet through transfer learning could offer a boost in classification performance by leveraging knowledge from larger datasets.
- **Exploring Transformer Networks:** Transformer networks, with their attention mechanisms and impressive performance on sequential data, could be explored for music classification. They may capture dependencies in audio data more effectively than traditional models.

Overall, the future direction of this research lies in refining feature extraction techniques, enhancing models' robustness through transfer learning, and expanding the scope to broader datasets and more sophisticated models.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning, 2016.

[2] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, 2002.

[3] Brian McFee, Colin Raffel, Dawen Liang, Daniel P.W. Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th Python in Science Conference*, pages 18–25, 2015.

[4] Andrada. Gtzan dataset - music genre classification, Mar 2020.

[5] Andradaolteanu. Work w/ audio data: Visualise, classify, recommend, Mar 2020.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[7] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16. ACM, August 2016.

[8] Aathreya S. Bhat, V.S. Amith, Namrata S. Prasad, and D. Murali Mohan. An efficient classification algorithm for music mood detection in western and hindi music using audio feature extraction. pages 359–364, 2014.

[9] Junghyun Kim, Seungjae Lee, and Won young Yoo. Implementation and analysis of mood-based music recommendation system. *2013 15th International Conference on Advanced Communications Technology (ICACT)*, pages 740–743, 2013.

[10] Jay Patel and E. S. Gopi. Musical notes identification using digital signal processing. *Procedia Computer Science*, 57:876–884, 2015.

[11] Keunwoo Choi, George Fazekas, and Mark Sandler. Automatic tagging using deep convolutional neural networks, 2016.

[12] K. Han, B. Kim, and S. Lee. Music genre classification using convolutional neural networks with support vector machine. *IEEE Access*, 6:20944–20951, 2018.

[13] J. Li, W. Wang, and Q. Zhang. Music genre classification with cnns and attention mechanism. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1–5, 2020.

[14] Simon Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[15] Jason Brownlee. How to build multi-layer perceptron neural network models with keras, Aug 2022.

[16] Multilayer perceptrons for digit recognition with core apis.

[17] H. Drucker and Y. Le Cun. Improving generalization performance using double backpropagation. *IEEE Transactions on Neural Networks*, 3(6):991–997, 1992.

[18] Anders Krogh and John A. Hertz. A simple weight decay can improve generalization. In *Neural Information Processing Systems*, 1991.

[19] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15:1929–1958, 2014.

[20] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, January 2015.

[21] Yann LeCun, Y. Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–44, 05 2015.

[22] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical Report ICS 8504, Institute for Cognitive Science, University of California, 1985.

[23] Michael I Jordan. Serial order: a parallel distributed processing approach. Technical Report ICS 8604, Institute for Cognitive Science, University of California, 1986.

[24] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

[25] Scott Zhu and Francois Chollet. Working with rnns, 2024. https://www.tensorflow.org/guide/keras/working$_w ith_r nns$.

[26] François Chollet et al. Keras. `https://keras.io`, 2015.

A Appendix

A.1 Hyperparameters Configuration

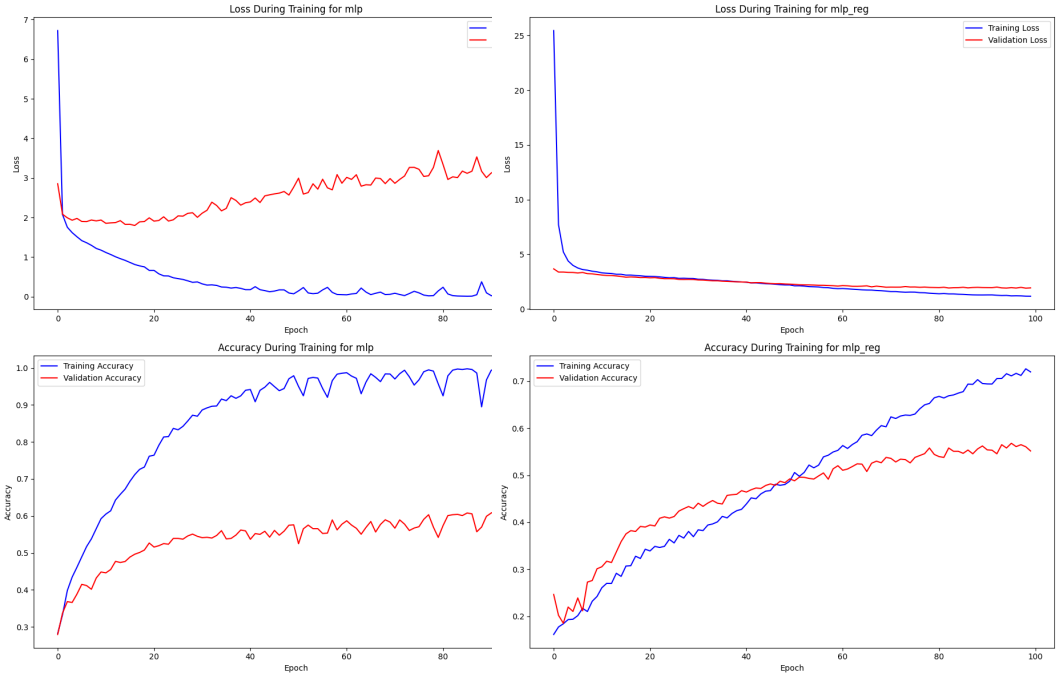
The hyperparameters used in our models are presented in Table 2. This configuration file, derived from `hparams.yaml`, specifies the parameters employed during the training of the models.

Table 2: Hyperparameters Configuration

Hyperparameter	**Value**
Common Parameters	
Sample Rate	22050
Duration (seconds)	30
Samples per Track	661500
Number of Segments	10
Number of MFCC	13
FFT Window Size	2048
Hop Length	512
Learning Rate	0.0001
Batch Size	32
Number of Epochs	100
Number of Classes	10
Activation Function	ReLU
RNN-Specific Parameters	
Architecture	"lstm"
LSTM Size	64
Dense Units	64
CNN-Specific Parameters	
Conv Layers	3
Filters	[32, 64, 128]
Kernel Sizes	[3, 3, 3]
Pool Sizes	[2, 2, 2]
Dense Units	64
Dropout Rate	0.3
Activation Function	ReLU
MLP-Specific Parameters	
Layers	[256, 128, 64]
Activation Function	ReLU
MLP_REG-Specific Parameters	
Layers	[256, 128, 64]
Dropout Rate	0.3
L2 Regularizer	0.001
Activation Function	ReLU

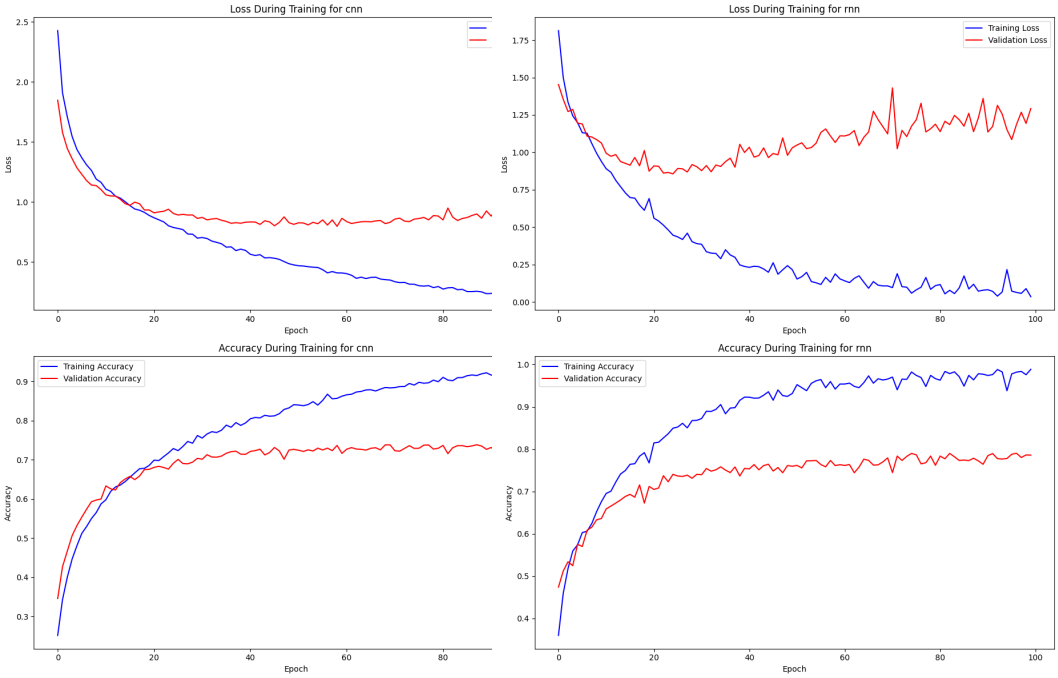
A.2 Training and Validation Curves

Figures illustrating the training and validation curves for each model are presented in Appendix A.2.



(a) MLP Training Curves

(b) MLP with Regularization



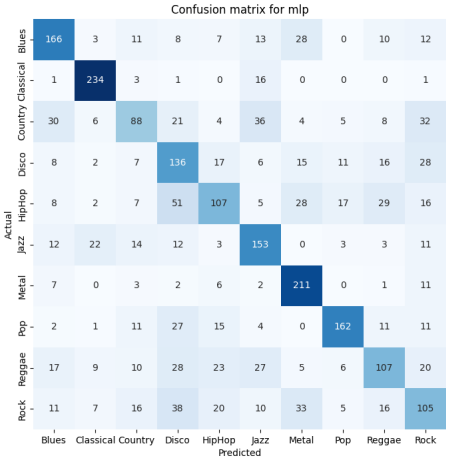
(c) CNN Training Curves

(d) RNN Training Curves

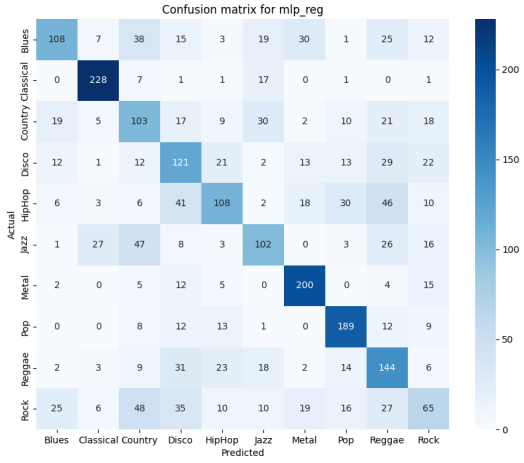
Figure 3: Training and validation curves for the models.

A.3 Confusion Matrices

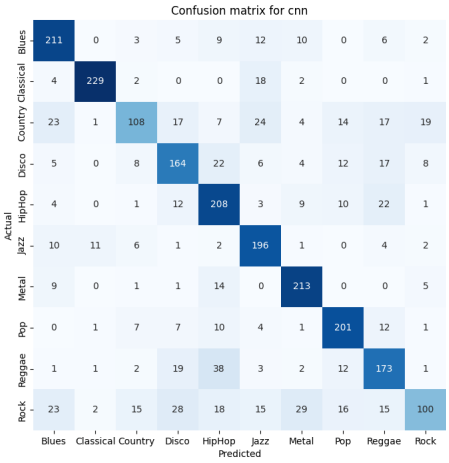
Figures illustrating the confusion matrices for each model are presented in Appendix A.3.



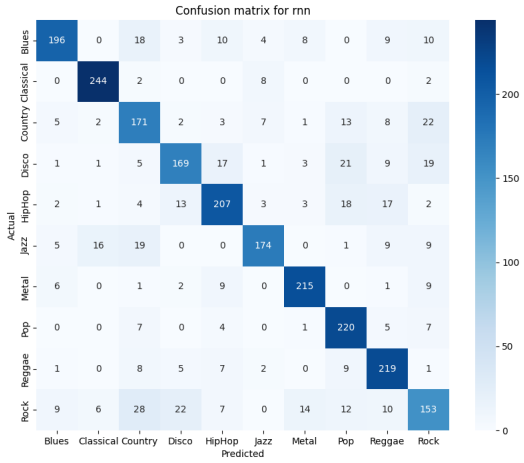
(a) MLP Confusion Matrix



(b) MLP with Regularization



(c) CNN Confusion Matrix



(d) RNN Confusion Matrix

Figure 4: Confusion matrices for the models.

A.4 Classification Reports

The classification reports in Table 3 provide a detailed analysis of the models' performance across each music genre. Each report includes the precision, recall, and F1-score for the respective genre classification model. These metrics help evaluate the effectiveness of the models in accurately classifying each genre.

Table 3: Classification Reports

(a) MLP Model

Class	**Precision**	**Recall**	**F1-Score**
Blues	0.63	0.64	0.64
Classical	0.82	0.91	0.86
Country	0.52	0.38	0.44
Disco	0.42	0.55	0.48
HipHop	0.53	0.40	0.45
Jazz	0.56	0.66	0.61
Metal	0.65	0.87	0.74
Pop	0.78	0.66	0.72
Reggae	0.53	0.42	0.47
Rock	0.43	0.40	0.41
Accuracy		0.59	
Macro Avg	0.59	0.59	0.58
Weighted Avg	0.59	0.59	0.58

(b) MLP_reg Model

Class	**Precision**	**Recall**	**F1-Score**
Blues	0.62	0.42	0.50
Classical	0.81	0.89	0.85
Country	0.36	0.44	0.40
Disco	0.41	0.49	0.45
HipHop	0.55	0.40	0.46
Jazz	0.51	0.44	0.47
Metal	0.70	0.82	0.76
Pop	0.68	0.77	0.73
Reggae	0.43	0.57	0.49
Rock	0.37	0.25	0.30
Accuracy		0.55	
Macro Avg	0.55	0.55	0.54
Weighted Avg	0.55	0.55	0.54

(c) CNN Model

Class	**Precision**	**Recall**	**F1-Score**
Blues	0.73	0.82	0.77
Classical	0.93	0.89	0.91
Country	0.71	0.46	0.56
Disco	0.65	0.67	0.66
HipHop	0.63	0.77	0.70
Jazz	0.70	0.84	0.76
Metal	0.77	0.88	0.82
Pop	0.76	0.82	0.79
Reggae	0.65	0.69	0.67
Rock	0.71	0.38	0.50
Accuracy		0.72	
Macro Avg	0.72	0.72	0.71
Weighted Avg	0.72	0.72	0.71

(d) RNN Model

Class	**Precision**	**Recall**	**F1-Score**
Blues	0.87	0.76	0.81
Classical	0.90	0.95	0.93
Country	0.65	0.73	0.69
Disco	0.78	0.69	0.73
HipHop	0.78	0.77	0.78
Jazz	0.87	0.75	0.81
Metal	0.88	0.88	0.88
Pop	0.75	0.90	0.82
Reggae	0.76	0.87	0.81
Rock	0.65	0.59	0.62
Accuracy		0.79	
Macro Avg	0.79	0.79	0.79
Weighted Avg	0.79	0.79	0.79