CS 4641 Team 4 Project Final Checkpoint

Introduction

For our project, we want to use supervised learning to create a machine-learning model that will categorize music audio into genres. We will use different pre-processing and machine-learning algorithms available with scikit-learn and other modules to compare performance.
Ali, et al. [1] found SVM to be the model to use when up against k-NN. Both Khammes, et al. findings [2] and Putra, et al. [3] look at the use of CNNs and RNNs and found better accuracy using them than other methods like SVM. Wu, et al. [4] use IndRNN, which lets each neuron process its hidden state independently over time, making it better for long-term data (e.g. 30-second samples in the GTZAN data).
We are using the GTZAN Genre Collection dataset at this link https://www.kaggle.com/datasets/carlthome/gtzan-genre-collection/data. The dataset consists of 30 second clips of .wav audio files categorized by music genres, with each file representing a different genre such as rock, jazz, and classical. Additionally, the dataset we found contained some data that was preprocessed into spectrograms which we can use for one of our models.

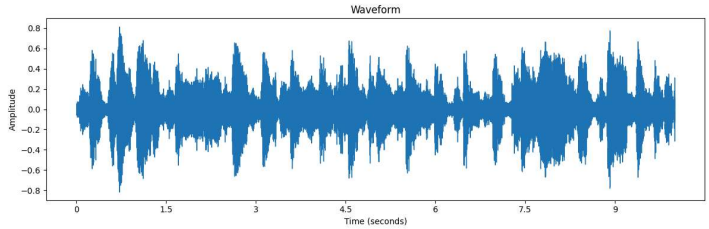
Problem Definition

We want to solve the issue of arranging short music clips into different genres, allowing for organized categorization. This automation reduces the need for manual classification, leading to cost-effective operations. The models will also remove human bias and subjectivity. Further development of these models will improve search, music discovery, and personalized recommendations. Pre-processing methods and machine-learning algorithms improve often even recent research quickly gets outdated in a few months. Our end goal is to improve on current classification models by using the greatest currently available open-source pre-processing and machine-learning algorithms.

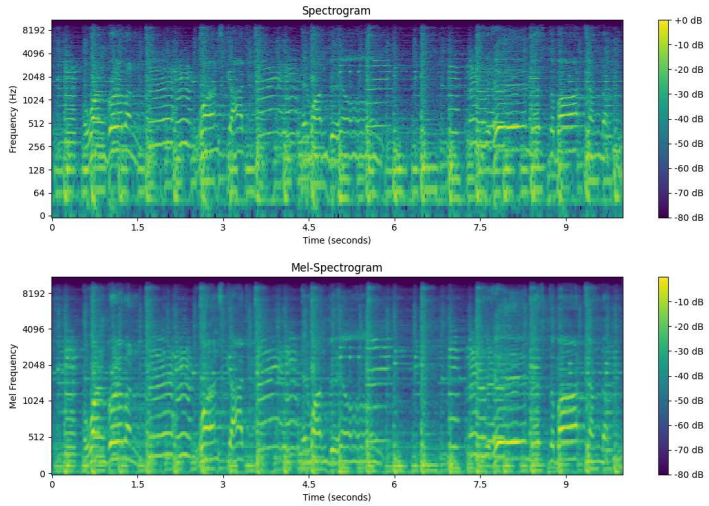
Methods

Preprocessing Methods

For our data we use the GTZAN Dataset which we found from our research papers to be a popular choice. It is split into two distinct sections, raw audio files and Mel spectrograms. We decided to ignore the pre-made Mel-spectrograms and make our own from the raw audio files. This was done so that we have more control over our data. First, we created helper functions to help us visualize the waveform of our raw audio. This helps us understand things like amplitude variations over time.

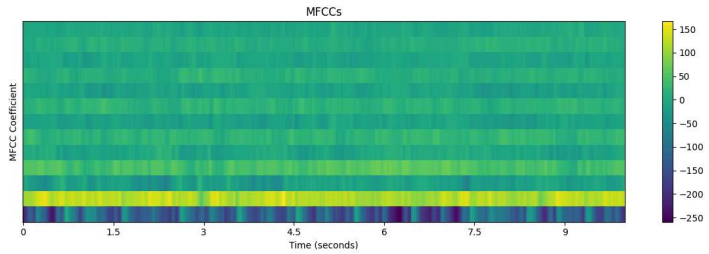


We then made a Mel-spectrogram helper function to visualize and understand the spectral characteristics of the audio. These Mel-spectrograms represent the short-term power spectrum of the audio signal on a Mel scale. This closely aligns with human auditory perception more than waveforms and regular spectrograms.



After exploring our different options for pre-processing, we decided on converting our audio into Mel-frequency cepstral coefficients (MFCCs). MFCCs are compact representations of the spectral envelope of an audio signal. These are more widely used in audio and

speech processing because they are good at detecting timbral aspects of audio. We feel like MFCCs can help us start with helpful features for our model.



Here are some parameters we set to uniformly process our files. Most of these are standard in audio processing.

```
SAMPLE_RATE = 22050
DURATION = 30
SAMPLES_PER_TRACK = SAMPLE_RATE * DURATION
N_MFCC = 13
N_FFT = 2048
HOP_LENGTH = 512
DATASET_PATH = "./Data/genres_original"
JSON_PATH = "data.json"
```

For our data we have 10 genres each with 100 audio files that are 30 seconds in length. Our data is labeled and organized in folders. We decided to divide all our audio into 3 segments of 10 seconds each. This was done to increase the size of our dataset, which ended up making our model more accurate due to the increased size of the data set.

We extracted the MFCCs from each segment and saved the data into a JSON file. The JSON object is called data and contains a mapping array that is a list of our genre labels (e.g., 'jazz', 'hip hop') and a labels array which is a list of all the labels for each segment. The labels are numeric (0-9) and correspond to our mapping array. We also have an MFCC array in data which is a list of MFCC arrays for each segment.

Some of our audio isn't over 30 seconds and are shorter than expected. We threw these files out because it didn't affect our data size in a major way. In the end, we ended up with 2970 data points to work with for our model.

We then combined our samples and labels arrays to create our X and Y matrices. After that, we split our data points up three ways: 80% for training data, 10% for validation data, and 10% for testing data.

```
Training set: (2376, 431, 13, 1), (2376,)
Validation set: (297, 431, 13, 1), (297,)
Test set: (297, 431, 13, 1), (297,)
```

These are the shapes for our new NumPy arrays. For each set, we print out the shape of X and then the shape of Y respectively. The last augmentation we do to our data is to flip the array along the time axis. We then add the flipped arrays as new data points. This effectively doubles the size of our original training set.

Finally, we save all our NumPy arrays as NumPy files to prepare them for our model.

Model Methods

Method 1: Convolutional Neural Network

For the actual model, we chose to go with implementing a Convoluted Neural Network (CNN). Our model is aimed at analyzing the features of the MFCC input data (the NumPy files, 2D arrays) and this is done by breaking the implementation down into 3 separate convolution blocks, and then adding a Global Pooling layer that is meant to reduce the parameter count (for manageability) and reduce overfitting. Finally, we did a dense layer with ReLU activation to ensure the model can do more complex audio combinations.

The reason we chose to use a CNN is because we found that convolutions can learn about small, local patterns within data, which, in our case, meant that it could be used to pick up on things like timbral textures, pitch, etc. By having layered convolutions, we can "incrementally" apply filters to learn more about the audios used over the short, 10 second time intervals. Our main goal is genre classification, so another benefit is that CNNs also provide translational invariance. This is useful because out of the hundreds of audio samples we used, having the capability to detect audio that has similarities (like being in the same genre) is easier to do since this ability comes with using CNNs. Another reason we chose this is because of the ability to stack layers - this means that by having the MFCC data pass through multiple convolution layers, we were able to hierarchically get more complex analyses of the arrays, leading to better accuracy.

```
layers.Conv2D(32, (3, 3), activation='relu', padding='same', input_shape=input_shape),
layers.BatchNormalization(),
layers.MaxPooling2D((2, 2)),
layers.Dropout(0.3),

layers.Conv2D(64, (3, 3), activation='relu', padding='same'),
layers.BatchNormalization(),
layers.MaxPooling2D((2, 2)),
layers.Dropout(0.3),

layers.Conv2D(64, (3, 3), activation='relu', padding='same'),
layers.BatchNormalization(),
layers.MaxPooling2D((2, 2)),
layers.Dropout(0.3),

layers.GlobalAveragePooling2D(),

layers.Dense(128, activation='relu'),
layers.Dropout(0.4),
layers.Dense(num_classes, activation='softmax')
```

In the first convolution layer, we applied 32 filters w/ a 3x3 kernel. We chose to go with ReLU activation as well since this allows for nonlinear learning while not compromising speed of learning. Then, we applied Batch Normalization to normalize the output followed by pooling to reduce the dimensions of the feature map (no overfitting). Finally we generalized it using a dropout of 30%. In the second layer, we upped it to 64 filters in order to find more niche/complex information from the data. The other properties were kept the same, so we were able to get deeper features extracted. The third block has the same settings as the second one, allowing us to get even more intricate information from the data.

We then did a global average pooling layer to reduce the feature maps we got out from the colvolutions. This works by calculating the average of all the values in the feature map and setting it to that number. This reduces overfitting while also keeping important information about the features. Finally, we did a dense layer to put all the features together for the final classification into the 10 genres we had. We also did a dropout of 40% to ensure we don't run into overfitting.

```
model.compile(
    optimizer=Adam(learning_rate=0.0001),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)
```

We also optimized when compiling the model, which was done using the Adam optimizer, with a learning rate of 0.0001. This is used to adjust the weights based on the loss gradient, which we found increased stability. For the loses function, we chose Sparse Categorical Crossentropy to make it work with integer labels (so we could do the classification). Finally, we recorded performance based on accuracy (training and validation).

Method 2: Recurrent Neural Network

We decided to choose RNNs because of its proficiency in modeling temporal sequences. RNNs can take advantage of MFCCs and understand the progression and patterns in our audio. RNNs memory structure also helps it retain context throughout audio and understand complex patterns within it.

First, we load our preprocessed data into training, validation, and test NumPy arrays. We then build our RNNs layers and compile it with an optimizer.

Layer (type)	Output Shape	Param #
gru (GRU)	(None, 431, 64)	15,168
dropout (Dropout)	(None, 431, 64)	0
batch_normalization (BatchNormalization)	(None, 431, 64)	256
gru_1 (GRU)	(None, 431, 96)	46,656
dropout_1 (Dropout)	(None, 431, 96)	0
batch_normalization_1 (BatchNormalization)	(None, 431, 96)	384
global_average_pooling1d (GlobalAveragePooling1D)	(None, 96)	0
dense (Dense)	(None, 128)	12,416
dropout_2 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 10)	1,290

We also use a monitoring techniques like early stopping to manage and control our training process.

```
early_stopping = EarlyStopping(
    monitor = 'val_loss',
    patience = 20,
    min_delta = 0
)
```

Method 3: SVM

For this model we decided to implement a Support Vector Machine. The reason we used an SVM for this task is because it can deal with high-dimensional, nonlinear data with relatively small datasets. Its flexibility, robustness, and interpretability make it a very good model for our music genre classification, offering a good trade-off between model complexity and generalization.

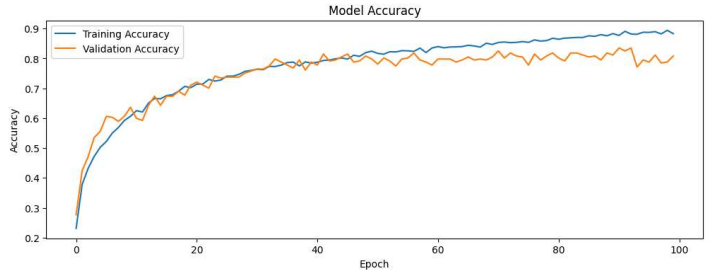
We had to add an extra preprocessing step of Flattening the data into vector format in order to meet the input requirements of the SVM model. After flattening each sample to a 2D format for compatibility with SVMs, the feature space expanded to 5,603 dimensions. We also applied StandardScaler to scale features, ensuring that each feature has zero mean and unit variance. This step is critical for SVM's performance, especially when using radial basis function (RBF) kernels.

```
svm_model = SVC(kernel='rbf', C=1.0, gamma='scale', probability=True)
```

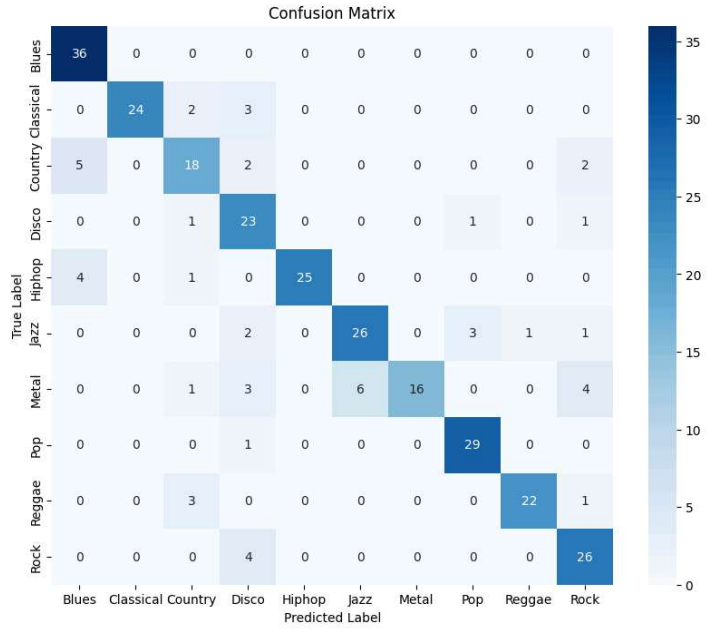
For the Model itself, we used had to define the Kernel, C, gamma, and probability as our hyperparameters. We decided to go with a Radial Basis Function (RBF) Kernel, which creates a decision boundary by considering the similarity between data point, allowing the model to make complex classifications effectively. Additionally, we decided to set C=1, which provides a balance between accuracy and generalization. We decided to set the gamma value to use the scale setting which automatically adjusts γ based on the number of features and the variance of the data. Instead of a binary class decision, we enabled Probability so that the model can return the likelihood that a sample belongs to each class.

• Results and Discussion

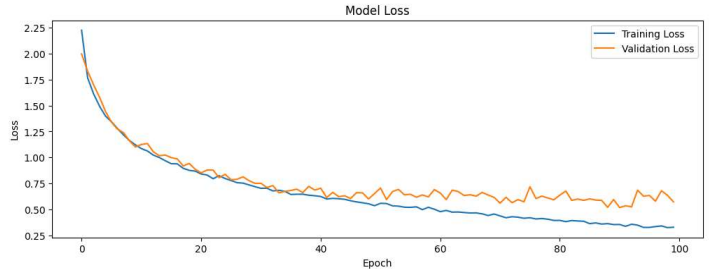
Method 1: Convolutional Neural Network



We ran our model for 100 epochs, and by the 100th epoch, our training accuracy reached around 90%. This is fairly accurate for an introductory implementation of our model, and this can likely improve with a few improvements. Additionally, the line in our graph shows rapid improvement in the early epochs and plateaus later on, indicating that our model is learning quickly at first and then learning less in the later epochs, which is to be expected. Since the gap between the training and validation accuracy is fairly minimal, we can conclude that overfitting is also minimal in our model.



Our confusion matrix shows that our model is performing fairly well. The main diagonal of the matrix is filled with values that are much larger than off diagonal cells, further indicating that our model is making correct predictions most of the time. However, this matrix gives more specifics on which genres are being misclassified. We can see that 14 out of 30 metal songs are being misclassified, with 6 of the misclassifications being jazz. We can also observe country being misclassified as blues 5 times. We observed some of these genres being musically similar, but jazz being misclassified as metal is a little concerning.



Over the 100 epochs we ran our model on, the training loss converged around 0.3. Our model is performing slightly worse on the unseen validation data (converging to a loss around 0.5), but this is fairly normal. Since the gap between the two lines is fairly minimal, we can conclude overfitting in our model is fairly minimal.

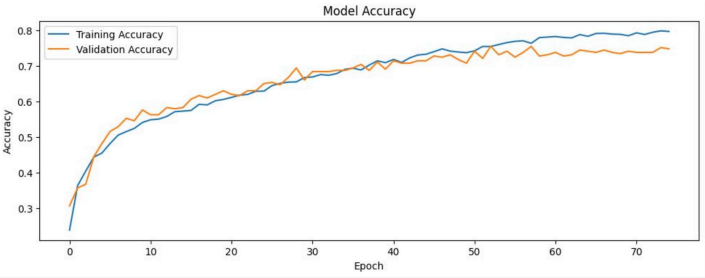
```
Test Accuracy: 82.49%
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 49ms/step
              precision    recall  f1-score   support

           0       0.80      1.00      0.89        36
           1       1.00      0.83      0.91        29
           2       0.69      0.67      0.68        27
           3       0.61      0.88      0.72        26
           4       1.00      0.83      0.91        30
           5       0.81      0.79      0.80        33
           6       1.00      0.53      0.70        30
           7       0.88      0.97      0.92        30
           8       0.96      0.85      0.90        26
           9       0.74      0.87      0.80        30

    accuracy                           0.82       297
   macro avg       0.85      0.82      0.82       297
weighted avg       0.85      0.82      0.82       297
```

We can observe from the stats above that we are achieving an average precision of 85% and an average recall of 82%. Overall, we're happy with these results for our initial iteration, however our support column has brought up an interesting potential issue with our model. Our total support value is 297, which is the number of samples in our dataset. This value is unsurprisingly divisible by 3, due to us splitting each song into 3 parts to increase the size of our dataset. However, some of the genres in our dataset do not have a support value divisible by 3, indicating our model is guessing parts of the same song to be different genres. This could be negatively affecting our results, and is another factor we are looking to improve in the future.

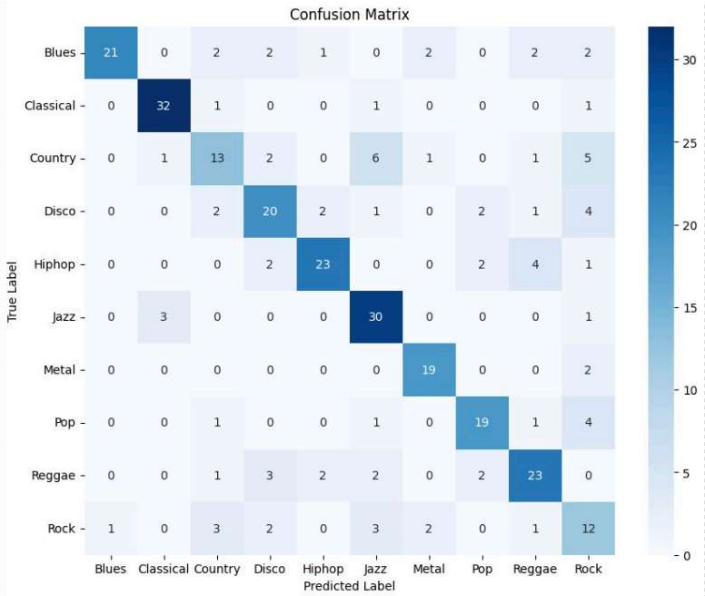
Our results are pretty good for right now, but there are a few interesting quirks with our results, that we are looking to improve on as we continue working on our project. For one, as I mentioned above we observed asymmetrical mispredictions in our confusion matrix. Looking at our graph, we are predicting a lot of metal songs as jazz, but not the other way around. This is something we are looking to

investigate and improve on in the future, but we are wondering if this is due to the sound quality of the songs in the dataset, or other issues. Additionally, we realized that when we split the songs into thirds, we are potentially putting some of one song in the testing data and the remainder of that song in the training data. This is something we are looking to fix moving forward, as we believe that could be causing some issues.

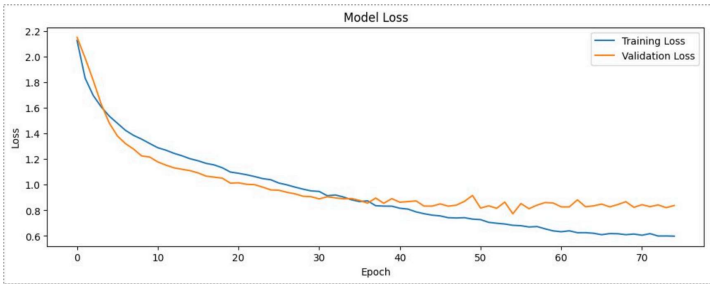
Method 2: Recurrent Neural Network



We ran our model for about 75 epochs, and by the 75th epoch, our training accuracy reached around 90%. This is fairly accurate for an introductory implementation of our model, and this can likely improve with a few improvements. Additionally, the line in our graph shows rapid improvement in the early epochs and plateaus later on, indicating that our model is learning quickly at first and then learning less in the later epochs, which is to be expected and implies that we would no benefit from many more epochs. Since the gap between the training and validation accuracy is fairly minimal, we can conclude that overfitting is also minimal in our model.



Our confusion matrix shows that our model is performing fairly well. The main diagonal of the matrix is filled with values that are much larger than off diagonal cells, further indicating that our model is making correct predictions most of the time. However, this matrix gives more specifics on which genres are being misclassified. We can see that 16 out of 29 country songs are being misclassified, with 6 of the misclassifications being jazz. We can also observe country being misclassified as rock 5 times. We observed some of these genres being musically similar and it is possible it has to do with the quality of our audio bytes.



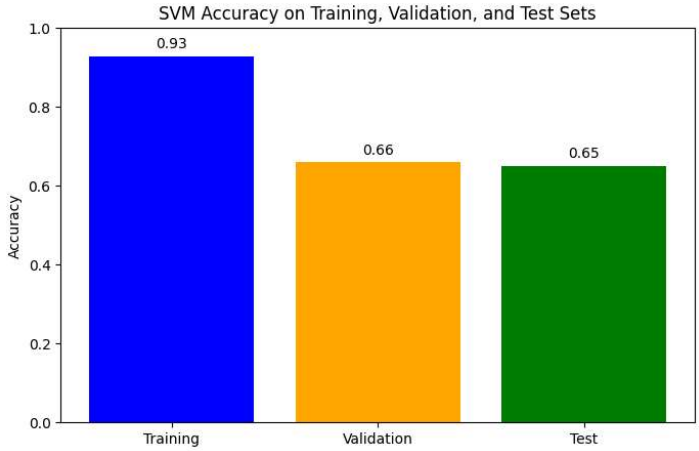
Over the 75 epochs we ran our model on, the training loss converged around 0.6. Our model is performing worse on the unseen validation data (converging to a loss around 0.9), but this is fairly normal. Since the gap between the two lines is fairly minimal, we can conclude overfitting in our model is fairly minimal.

```
Test Accuracy: 71.14%
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 219ms/step
              precision    recall  f1-score   support

           0       0.95      0.66      0.78        32
           1       0.89      0.91      0.90        35
           2       0.57      0.45      0.50        29
           3       0.65      0.62      0.63        32
           4       0.82      0.72      0.77        32
           5       0.68      0.88      0.77        34
           6       0.79      0.90      0.84        21
           7       0.76      0.73      0.75        26
           8       0.70      0.70      0.70        33
           9       0.38      0.50      0.43        24

    accuracy                           0.71       298
   macro avg       0.72      0.71      0.71       298
weighted avg       0.73      0.71      0.71       298
```

We can observe from the stats above that we are achieving an average precision of 72% and an average recall of 71%. Overall, we're happy with these results for our initial iteration, however our support column has brought up an interesting potential issue with our model. Our total support value is 297, which is the number of samples in our dataset. This value is unsurprisingly divisible by 3, due to us splitting each song into 3 parts to increase the size of our dataset. However, some of the genres in our dataset do not have a support value divisible by 3, indicating our model is guessing parts of the same song to be different genres. This could be negatively affecting our results, and is another factor we are looking to improve in the future and is an issue we also noticed in our CNN.

Overall, these results are very good and so far, are analogous with our research that wrote that Neural Networks are the best tool for classifying music in this fashion. One issue we noticed particularly with this model is its problem classifying rock and country music. We believe that this can be improved by adding more complexity to our model.

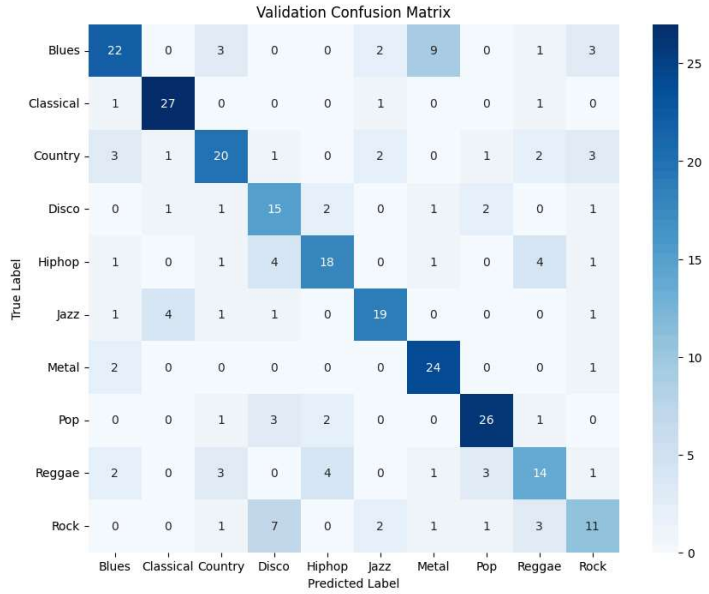
Method 3: SVM

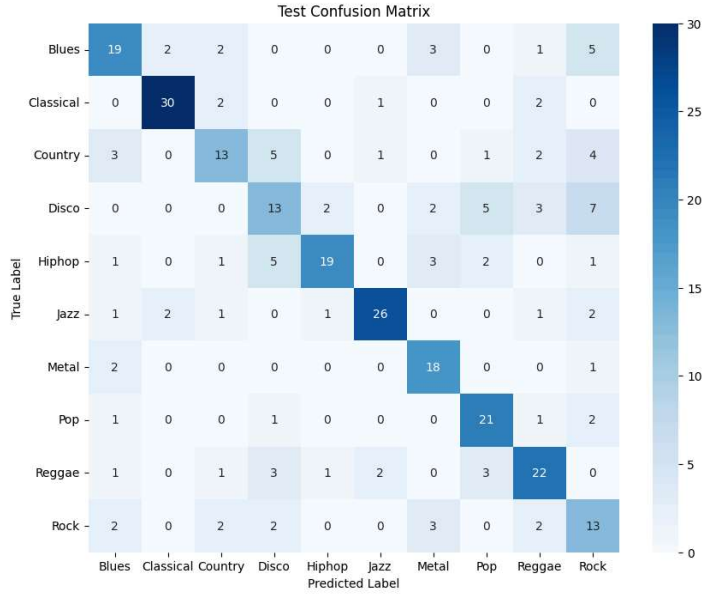


The training accuracy of 93% shows that the model can fit the training data very well, effectively capturing the patterns in the training samples. However, such a high training accuracy raises concerns about overfitting. The model might be focusing too much on the training data, failing to generalize to unseen examples. The Validation Accuracy of 66% shows a drop that suggests that the model struggles to generalize to unseen data. This discrepancy between training and validation accuracies is a key indicator of overfitting. The test accuracy (65%) being close to the validation accuracy shows that the model's performance on unseen data is consistent.However, a test accuracy of 65% is modest, indicating that the SVM model has difficulty separating some genres or handling the overlap in their audio features.

```
Training Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.93      0.94       456
           1       0.95      0.99      0.97       458
           2       0.91      0.92      0.91       458
           3       0.88      0.88      0.88       484
           4       0.93      0.89      0.91       464
           5       0.96      0.93      0.94       478
           6       0.91      0.96      0.93       504
           7       0.96      0.97      0.96       482
           8       0.94      0.92      0.93       478
           9       0.89      0.89      0.89       494

    accuracy                           0.93      4756
   macro avg       0.93      0.93      0.93      4756
weighted avg       0.93      0.93      0.93      4756

Validation Classification Report:
              precision    recall  f1-score   support

           0       0.69      0.55      0.61        40
           1       0.82      0.90      0.86        30
           2       0.65      0.61      0.62        33
           3       0.48      0.65      0.56        23
           4       0.69      0.60      0.64        30
           5       0.73      0.70      0.72        27
           6       0.65      0.89      0.75        27
           7       0.79      0.79      0.79        33
           8       0.54      0.50      0.52        28
           9       0.50      0.42      0.46        26

    accuracy                           0.66       297
   macro avg       0.65      0.66      0.65       297
weighted avg       0.66      0.66      0.66       297

Test Classification Report:
              precision    recall  f1-score   support

           0       0.63      0.59      0.61        32
           1       0.88      0.86      0.87        35
           2       0.59      0.45      0.51        29
           3       0.45      0.41      0.43        32
           4       0.83      0.59      0.69        32
           5       0.87      0.76      0.81        34
           6       0.62      0.86      0.72        21
           7       0.66      0.81      0.72        26
           8       0.65      0.67      0.66        33
           9       0.37      0.54      0.44        24

    accuracy                           0.65       298
   macro avg       0.65      0.65      0.65       298
weighted avg       0.67      0.65      0.65       298
```

The training classification report shows consistently high precision, recall, and F1-scores (all above 0.89), indicating the model has effectively learned the training data patterns. However, this near-perfect performance suggests the model has likely overfit the training data, memorizing specific characteristics rather than learning generalized trends. Overfitting is further supported by the significant drop in performance on the validation and test sets, indicating the model struggles to generalize to unseen examples. This imbalance between training and test performance reflects the challenges of working with high-dimensional audio data and the limited ability of SVM to handle overlapping or nuanced features in certain genres. The validation and test classification reports highlight a moderate weighted F1-score of 0.66, showing the model performs inconsistently across genres. High-performing genres such as Classical (F1-score 0.86) and Metal (F1-score 0.75) demonstrate the model's ability to handle distinct audio patterns, likely due to their unique feature characteristics. In contrast, genres like Disco (F1-score 0.56), Rock (F1-score 0.46), and Reggae (F1-score 0.50) perform poorly, reflecting significant overlap in feature representation with other genres. For example, Disco shares rhythmic similarities with Hiphop and Pop, leading to frequent misclassification. These discrepancies highlight the limitations of the current feature set and the need for more sophisticated feature extraction techniques to improve genre separability.





In the confusion matrix of validation, Classical, Metal, and Pop had the best results with correct predictions of 27/30, 24/27, and 26/33 samples. This result indicates that such genres contain distinct and well-separated features that the SVM model could efficiently distinguish. For instance, there might be harmonic patterns and frequencies of a range that may typically define Classical music, whereas Metal could be associated with a high energy and loudness with which the model would more easily grasp its nuances, while Pop is slightly less distinctive; hence, its rhythmic and tonal features are moderately typical, and thus above average. However, the matrix also revealed very significant challenges in the classification of overlapping genres like Disco, Rock, and Reggae. Disco had correctly predicted 15 out of 23 samples and tends to get misclassified as either Pop or Rock, probably due to the shared rhythmic and energetic traits. Similarly, Rock has weak diagonal dominance, with only 11 correct predictions out of 26, and is often misclassified as Reggae, Disco, or Metal, reflecting the overlap in instrumental and tonal features across these genres. Also, Reggae has only 14 correct predictions out of 28, struggling with confusion against genres like Hiphop and Rock, where shared tempos or instrumental characteristics may lead to ambiguity. These misclassifications emphasize that the current feature set poorly characterizes the subtle variations between these overlapping genres and needs further refinement.

The test confusion matrix reinforces the patterns in the validation matrix, showing consistent trends in performance across genres. For example, 'Classical' is still very accurate: 30 out of 35 samples were correctly identified. The consistency would suggest that features defining Classical are robust and generalize well to unseen data. Jazz, with 26 correct out of 34, performs even better on the validation set, which indicates this genre is well learned by the model in terms of features that distinguish it from others. This would be the same with regards to Pop: 21 out of 26; genres for which it gets mostly wrong are Disco and Rock, which confirms, in the feature space, that this genre is relatively distinctive. However, the test matrix also brings out the fact that there are inalienable problems with genres such as Disco, Rock, and Reggae, where the performance of the model remains poor. For instance, Disco, rightly predicted only 13 out of 32, is still largely misclassified as Rock or Reggae because of some shared rhythmic and tonal characteristics. Among them, Rock has only 13 correct predictions out of 24, showing its significant confusion with genres like Pop and Reggae because it is not distinctive in the feature set. Reggae, on the other hand, shows slight improvement with 22 correct predictions out of 33 but is still struggling due to overlaps in tempo and tonal features with Hiphop and Rock. These trends indicate that the model generalizes well for very distinctive genres, but for similar genres, this remains a real challenge. Such problems will be solved by improvements in feature extraction techniques and by adding more data for underrepresented genres to learn more about their unique features.

Our SVM model demonstrated strong performance on the training data, but its significant drop in accuracy and F1-scores on the validation and test sets revealed that it may not be the most suitable model for this task. While we were able to effectively capture patterns for distinct genres like Classical and Metal, the SVM struggled to handle overlapping genres such as Disco, Rock, and Reggae, where shared rhythmic and tonal characteristics led to frequent misclassifications. This suggests that the SVM's reliance on linear separability and limited feature space transformations was insufficient for the complexity and nuance of our audio data. We have concluded that a more sophisticated model, such as a neural network capable of capturing temporal and spectral dependencies, would likely perform better for this problem.

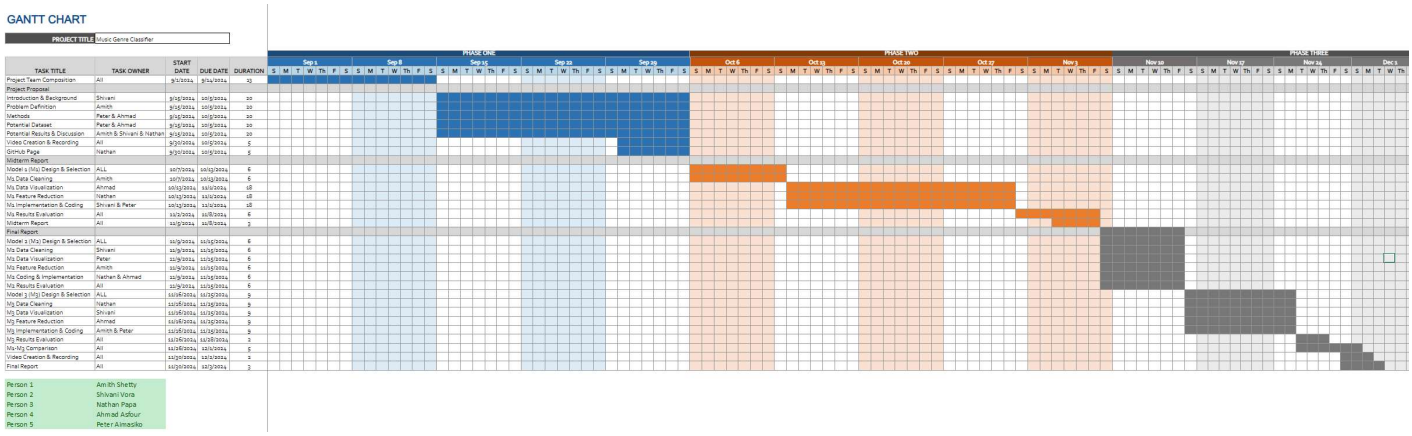
Comparison

On the whole, our CNN model performed the best out of the three models we implemented. It had the highest accuracy, lowest loss, best confusion matrix results, and all while having a fairly low training time. Our RNN had the second best results, with a still decent accuracy of around 71% and a reasonably low loss. However, the main place where our RNN fell short was in its training time which took around 90 minutes compared to the other two taking less than 10. This proves that in this case, the additional context that RNNs capture is not producing any better results than a CNN was, and just makes the training take way longer. Our SVM model had the worst results, with an accuracy in the mid 60s and the highest loss of the three models. We conclude this is likely to do with the fact that we had to normalize and flatten the data for the SVM which likely caused us to lose some of the data's complexity, making it harder to classify.

References

- [1] M. A. Ali and Z. A. Siddiqui, Automatic Music Genres Classification Using Machine Learning, https://pdfs.semanticscholar.org/e272/ce9dd8c7adec73817135d053f5819d968082.pdf (accessed Oct. 3, 2024).
- [2] A. Khamees, H. Hejazi, M. T. Aishurideh, and S. A. Salloum, Classifying Audio Music Genres using CNN and RNN, https://www.researchgate.net/publication/349804849_Classifying_Audio_Music_Genres_Using_CNN_and_RNN (accessed Oct. 3, 2024).
- [3] P. H. Putra, K. M. Tay, and M. Isa, "A Review of Feature Extraction and Classification Algorithms for Music Genre Recognition," Applied Sciences, vol. 8, no. 1, p. 150, Jan. 2018, doi: 10.3390/app8010150.
- [4] W. Wu, F. Han, G. Song, and Z. Wang, "Music Genre Classification Using Independent Recurrent Neural Network," 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1-5, 2018, doi: 10.1109/ICASSP.2018.8461749.

Gantt Chart



Picture of Gantt Chart above because it's easier for printing to PDF.

Scroll Up and Left if spreadsheet is not visible
Link to Gantt Chart if not visible above (may cause an issue when signed into Office365, try incognito mode): Gantt Chart Link

- ## Contribution Table

Member	Contribution
Nathan Papa	Model Design, Feature Reduction, Results, and Report
Shivani Vora	Model Design, Implementation, Results, and Report
Amith Shetty	Model Selection, Data Cleaning, Results, and Report
Peter Aimasiko	Model Design, Implementation, Results, and Report
Ahmad Asfour	Model Selection, Data Visualization, Results, and Report