IDS 705 Final Report

Genre Classification of Music Audio Using Deep Learning

A Comparison of Model Training Using Spectrograms and Waveforms

Team Members:
Jennifer Li
Javidan Karimli
Jahnavi Maddhuri
Rishika Randev

Team Identifier: J3R Beats (Dolphin)

Abstract

Our project investigates multiple strategies for improving music genre classification using deep learning models. We used the GTZAN Genre Collection dataset of 1,000 audio samples across 10 genres, trained and evaluated it with a variety of models in four experiments—with a focus on evaluating how different input representations (spectrograms, raw waveforms, and pre-extracted musical features), model architectures (custom and pre-trained), data augmentation techniques, and training set sizes affect performance. Our experiments revealed that models trained on pre-extracted musical features, particularly XGBoost (Tuned) achieved the highest accuracy (79%) compared to spectrogram-based models (best: 76% with VGG16) and the fine-tuned raw waveform models (68%).These findings suggest that carefully extracted musical features might remain competitive with and even outperform deep learning approaches based on spectrograms or raw waveforms for music genre classification tasks. Our project contributes to the development of more accurate and scalable genre classification systems in music streaming and audio recognition.

Introduction

Music plays a vital role in human culture, expression, and emotion, allowing for the communication of common experiences and stories that transcend time and geography. As the global music industry grows and digital platforms dominate how we access and engage with music, the ability to automatically classify music by genre has become increasingly important. Accurate genre classification helps drive personalized recommendations, organize vast music libraries, and improve user experience in music streaming services. Music genre refers to a category that identifies pieces of music as belonging to a shared tradition or set of conventions, often based on musical characteristics such as rhythm, instrumentation, form, and cultural context (Tzanetakis & Cook, 2002). Automatically distinguishing between music genres poses significant challenges due to the inherent subjectivity of genre definitions, overlaps between styles, and the complex nature of audio signals.

Our project focuses on comparing three different approaches to training machine learning models for music genre classification: using spectrograms, raw waveforms, and pre-extracted musical features as input data. Spectrograms provide a visual representation of audio frequencies over time, capturing rich temporal and spectral patterns, while raw waveforms retain the complete, unprocessed audio signal. Pre-extracted musical features leverage domain knowledge to capture specific audio characteristics like tempo, harmonic-percussive separation, and chroma features. The central question is: which input representation enables more accurate and efficient genre classification? By comparing these three approaches with both custom and pre-trained model architectures, we aim to uncover which combination leads to better model performance and is more effective and scalable, as well as the potential reasons behind it.

This project not only pushes us to think critically about how machines "hear" and interpret music, but also has real-world applications in improving audio recognition, which is both technically interesting to data scientists and practically relevant to audio recommendation systems and subsequently, the listener's experience. Eventually, accurate genre classification and understanding of genre similarities could help music streaming apps to better recommend content to its listeners.

The goal of this project is to determine which input representation—spectrograms, raw waveforms, or pre-extracted musical features—yields better performance in training machine learning models for music genre classification, while also investigating the impact of model architecture selection, data augmentation techniques, and training dataset size on classification accuracy. Different modeling techniques will be used for each input representation based on what is appropriate to the input form, and proven to have strong performance in previous research studies.

Background

Several studies have explored music genre classification using deep learning, focusing on different input representations and model architectures. One approach is to first extract relevant features, such as density, tone, and other acoustic qualities, from audio samples and then train genre classification models. Other studies have used spectrograms as input; for example, Costa et al. (2011) converted audio signals into spectrograms and treated them as texture images for classification, showing the effectiveness of visual audio features. Patil et al. (2023) used spectrograms and a custom deep neural network to classify Western and Indian genres, demonstrating improved accuracy on several datasets. Zhang and Li proposed three CNN models, known as parallel CNN ensembles, to provide a detailed pattern for expressing musical artistic attributes (2025). In addition, Er et al. (2019) focused on the related-tasks of music mood recognition using chroma spectrograms, highlighting the versatility of this representation.

While spectrogram-based models are well-studied, fewer works have compared them directly with waveform-based models and feature-based approaches under the same conditions. In Chang et al.'s 2021 paper, a SincNet implementation, which uses a raw audio file as input, was compared with MS-SincResNet, which uses the raw audio file and spectrograms of the same audio files to classify the genre. More recently, Kim et al. (2023) explored parameter-efficient tuning methods for Wav2Vec2 models on raw waveform inputs. However, comprehensive comparisons across all three input types remain limited in the literature.

Therefore, we aim to extend previous work by conducting a controlled comparison between three input types - spectrograms, raw waveforms, and pre-extracted musical features - using both custom and pre-trained model architectures for each input type. Through this comprehensive comparison, we aim to determine which representation is most effective for determining genre, therefore contributing new insights to the field and helping guide future model design.

Data

The main source of data in this project was the GTZAN Genre Collection, originally collected and used in the paper *Musical Genre Classification of Audio Signals* by G. Tzanetakis and P. Cook (2002). This dataset contains 100 30 second audio files across 10 different genres, for a total of 1000 samples. The genres in this dataset include blues, classical, country, disco, hip hop, jazz, metal, pop, reggae and rock (Thome, 2020). These audio samples were collected from various sources between 2000 and 2001, including CDs, radios, and microphone recordings. We converted these samples into both 1) raw waveform time series data, which captures amplitude information over time, and 2) spectrograms, which are built from taking a discrete Fourier transform of multiple small segments of the audio, and then stacking them together see how frequency and amplitude change over time. We specifically generated Mel spectrograms, which are a filtered version of a regular spectrogram where frequencies are represented on the mel scale. This scale more accurately reflects how human listeners perceive frequency, which is different from actual pitch because humans are able to distinguish changes in lower frequencies more easily than changes in higher frequencies (Hugging Face, n.d.).

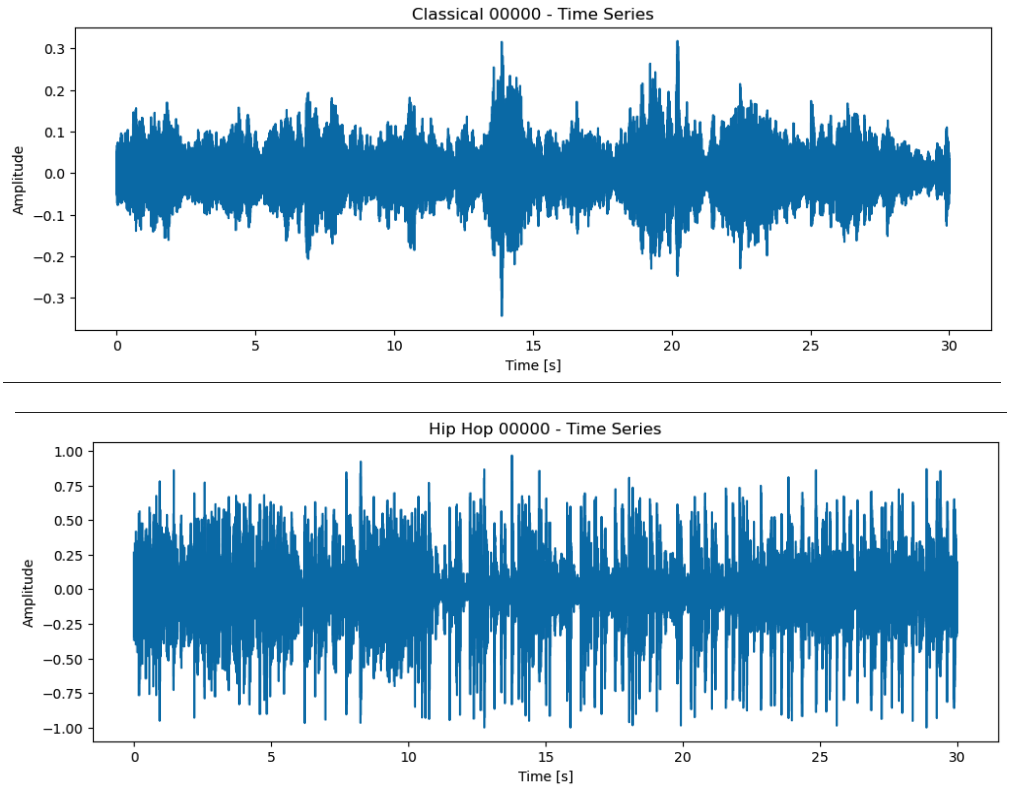

Figure 1. These two plots depict raw waveform (timeseries) data for a classical audio sample (top) and hip hop sample (bottom).

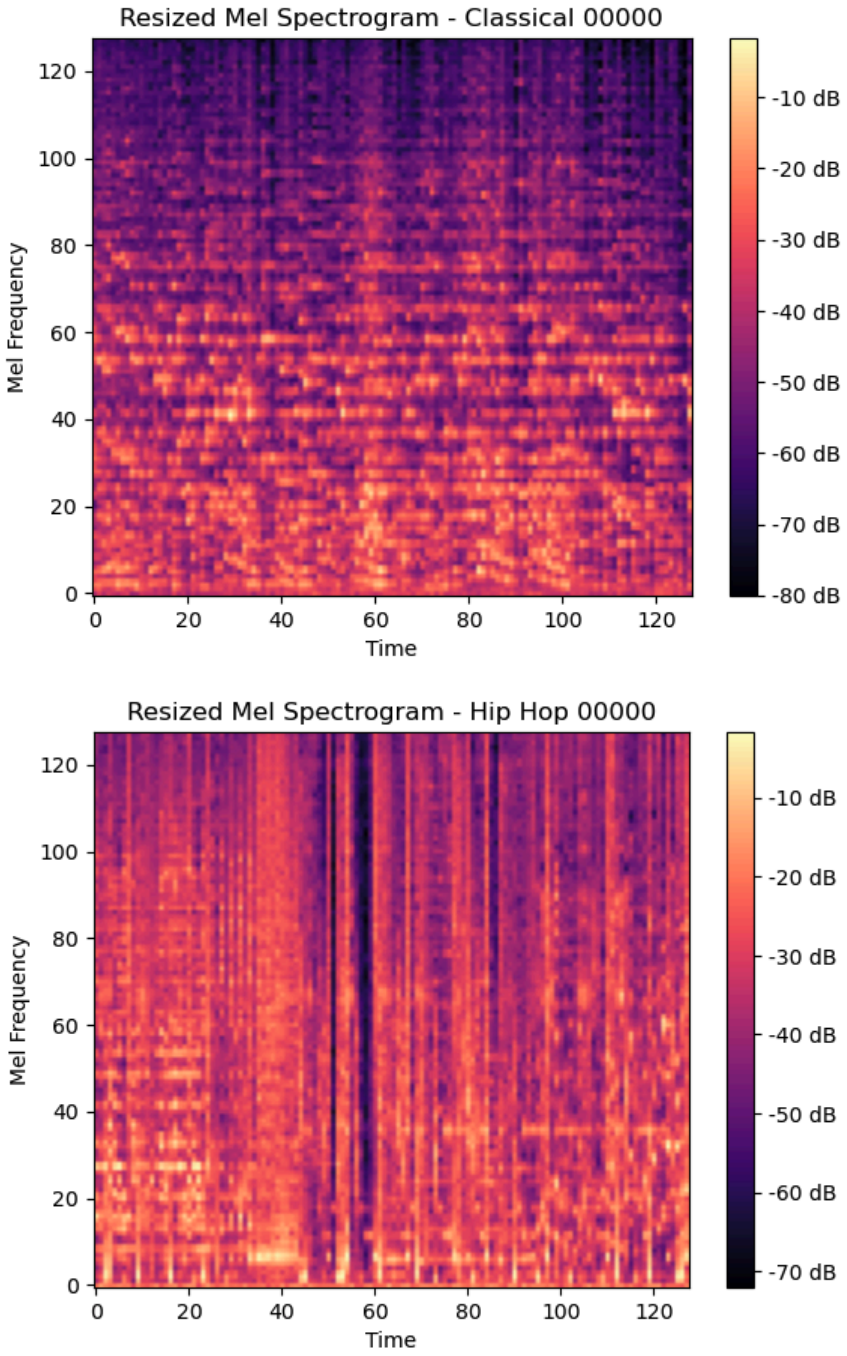

Figure 2. This figure depicts Mel spectrograms (after being resized to be 123x123 pixels) of the same audio files that are shown as waveforms in Figure 1.

The Kaggle collection we accessed also included a csv of pre-extracted features of these audio files. We used these three different representations to train various models in Google Colab using a T4 GPU. An 80-20 train-test split with the same seed was used across all models to ensure comparability of generalization performance; however, for Experiment 2, the split was done prior to augmentation (to ensure that only the training data was augmented), meaning the final ratio between training and test was not actually 80-20.

Experiments

Preprocessing

For Experiments 1 and 2, the Python Librosa library was used to generate Mel spectrograms from the audio files; the numerical pixel data from these spectrograms was resized and RGB-mapped to be 128x128x3 (where 128 x 128 represents each of the pixels and 3 represents the RGB scale of that pixel), and then normalized so that they were ready to use as input for our custom and pre-trained spectrogram-based models.

For the remaining experiments, the .wav audio files as well as the pre-extracted musical feature datasets were readily available through Kaggle and extensive data preprocessing was not needed.

Experiment 1: Spectrogram Custom vs. pre-trained Model Comparison

Custom Model

While developing a custom model to train on the numerical matrix representation of the spectrogram, CNN and CRNN models were used. In their studies, Meng (2024) and Er and Aydilek, (2019) mention the success of using CNN models for this type of classification. Meng achieved an accuracy of 91% in his CNN model when training based on 3-second frames. Er et al. also used CNN as their baseline model. Apart from CNN, CRNN model was also tried with an LSTM layer as the final layer in the model. The hypothesis was that CRNN would be better at capturing the temporal structure of the spectrogram and better be able to identify the patterns within the data. During the tuning process, Batch normalization, layer count, filter sizes, pooling techniques, bidirectional LSTMs, regularization and early stopping were all considered and implemented to attain the best model for both the CNN and CRNN.

Pre-Trained Models

Three pre-trained models that are traditionally used with image input were fine-tuned using the spectrogram files: AlexNet, VGG16, and ResNet. AlexNet was originally trained to distinguish 1000 different image classes, and consists of five convolutional layers plus three fully connected layers, with ReLu used as the activation function and dropout used for the first two fully connected layers. VGG16 is a similar deep convolutional neural network but includes a larger number of convolutional layers (13), as well as a fixed smaller filter size across convolutional layers (3x3), and was able to achieve a higher accuracy on the ImageNet database than AlexNet (Kurama, 2024). ResNet, introduced in 2015 in attempts to avoid the vanishing gradient problem observed in very deep neural networks, was able to improve on both models by making use of a residual network architecture, where skip connections allow input to flow as it is between some layers. While multiple layer depths can be used in ResNet, we opted to use ResNet18 (18 layers) because the relatively small number of samples that we had did not warrant a very deep network (Residual Networks - Deep Learning, 2025).

Taking inspiration from Er et al.'s (2019) AlexNet and VGG16 fine-tuning methodology for spectrogram input, we experimented with different approaches, including both training only the last layers of the networks and unfreezing and training all of the layers' weights. The former involved training models with either the first, second, or third fully connected layer as the final classification layer, effectively using some subset of the original models as a feature extractor. Additional details of this experimentation are provided in Appendix A.

Experiment 2: Augmented Spectrogram Inputs for Classification

For this experiment, we performed data augmentation on the training dataset. The goal of this experiment was to see how much performance improvement could be derived from both increasing the size of our training data and allowing our spectrogram-based models to see a greater variety in the data and thus generalize genre patterns more effectively. We used Librosa and numpy to augment the audio files in the training set, increasing our training size sixfold, and then created spectrograms from them. For each audio file, we performed time stretches by four different factors (0.81, 0.93, 1.07 and 1.23), and also performed a time shift by 5 seconds, based on work by Er & Aydilek (2019). We then trained all of our models for Experiment 1 (both custom and pre-trained) using this new augmented training set, maintaining the same test set as in all other experiments.

Experiment 3: Raw Waveform Inputs Using Pre-Trained Model vs. Extracted Features Inputs Using Custom Models

This experiment compares two different inputs for genre classification: raw waveform (timeseries) data vs. pre-extracted feature data.

Raw Waveform Inputs Using Pre-Trained Model

In this experiment, we used Facebook's Wav2Vec2-base model to learn audio representations directly from raw waveforms. Wav2Vec2-base model is a self-supervised deep learning architecture introduced by Baevski et al. (2020) and developed by Facebook AI (now Meta AI). It can significantly improve upon its predecessor by combining a multi-layer convolutional feature encoder with a Transformer network, allowing it to model long-range dependencies in audio. It has been successfully adapted to tasks like speech emotion recognition (Shon et al., 2021) and genre classification (Wang et al., 2022).

Inspired by these applications, we designed two different experiments, to see if the improvements to model architecture and training strategies can enhance music genre classification performance:

- For the first experiment, we processed the music clips sampled at 16kHz, ran them through the Wav2Vec2 model to extract features, and then passed these features into a basic neural network with two linear layers (768→128→10) separated by ReLU activations and

dropout (0.3). We didn't fine-tune the main Wav2Vec2 part (keeping it "frozen") and used a steady learning rate of le-3, trained for 10 epochs, throughout training.

- For the second Experiment, we enhanced our approach with a more sophisticated architecture: we added extra normalization layers so now we used a three-layer classification head (768→256→128→10), varied the dropout rates (0.3, 0.2), reduced learning rate (le-4/10), trained for 15 epochs. Also, we used a progressive training strategy where we initially froze the encoder but unfroze it after 5 epochs, to fine-tune the entire system.

Musical Features

The GTZAN dataset also includes various musical features captured for each wav file using the python library Librosa, such as zero-crossing rate to show how noisy the track is, harmonic-percussive separation to split tone from rhythm, tempo in beats per minute to capture the song's speed, spectral centroid and roll-off to indicate brightness and where most energy lies, MFCCs to summarize the overall tone shape, and chroma features to reflect the underlying chords. It is worth noting that most of these features produce a sequence of numbers one value per analysis frame (Librosa splits the audio into overlapping windows based on the sample rate and hop length). To capture how they change over time, we typically summarize each feature's mean and variation rather than use every individual frame value. Since these features span different scales and units, we applied standard scaling to normalize them and boost performance. The scaler was fitted on the training set and then used to transform the test set as well.

Experiment 4: Train and Test Split Comparison

After selecting the best performing model from the previous experiments, we conducted two targeted analyses. First, we applied bootstrap sampling to assess how minor variations in training data composition influenced performance. Next, we varied the overall size of the training set to pinpoint where accuracy began to plateau. This was done because it is often hard to obtain a large number of audio samples from different genres–and knowing how many samples are sufficient for genre classification can be beneficial for the future. To conduct this experiment, nine different train sizes and bootstrapping on the training data were used, and the accuracy of the resulting models was compared to determine at what training size performance seemed to level off.

Figure 3 below summarizes the overall experimental design and models used for different input types.

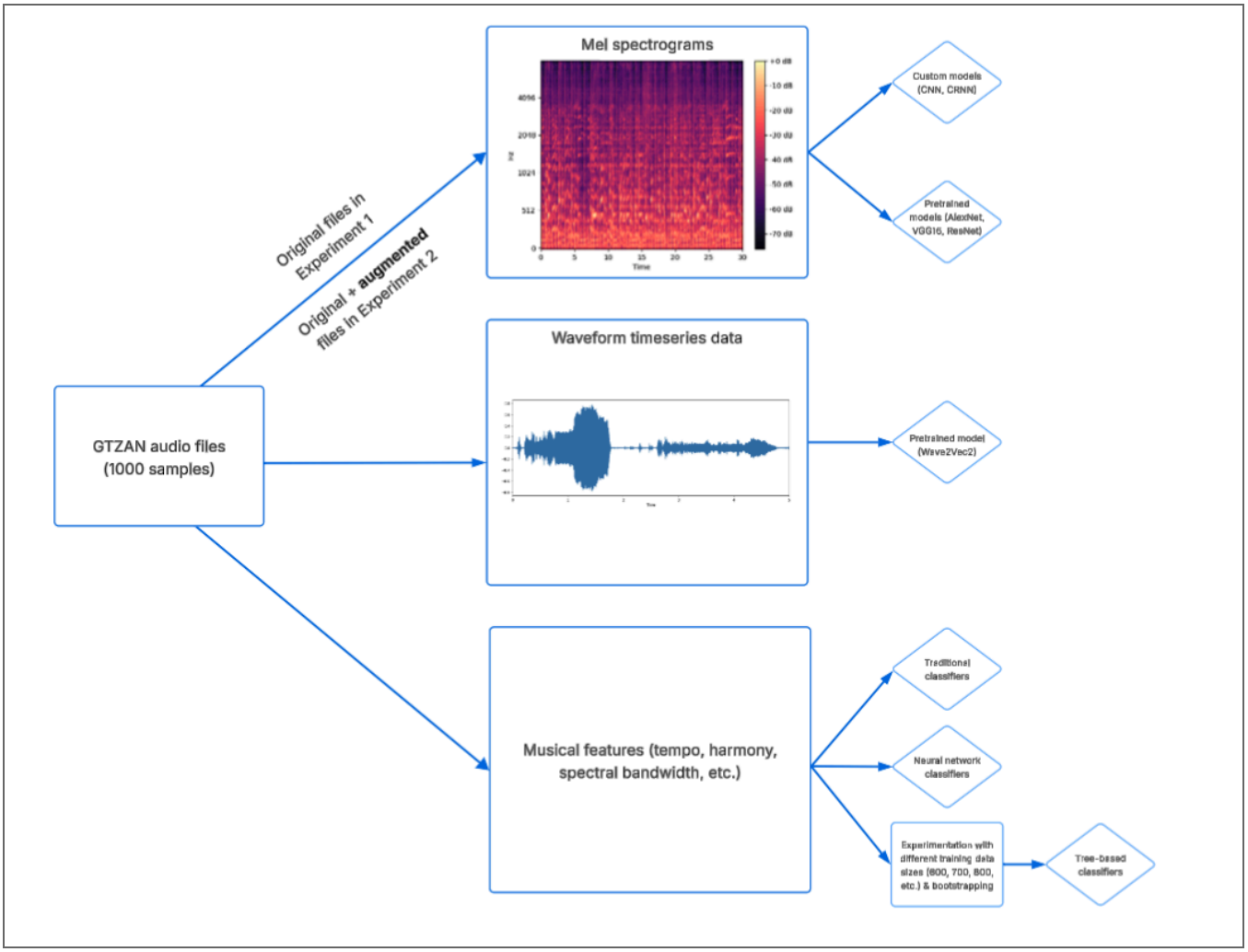

Figure 3. This flowchart depicts the overall methodology of the project. The original audio files were represented in three different input forms (spectrogram, raw waveform, and extracted musical features) and subsequently used to train custom and/or pre-trained models suitable for the input type.

Model Evaluation

Generalization performance was evaluated using a held out test set of 200 samples; a validation set was not used to maximize the number of samples available for training, which was especially important for use with the pre-trained deep networks. Accuracy was the main measure of performance that we assessed on the test set, as the data was balanced among the 10 genres / classes; it is also the primary metric reported in past studies that used spectrograms and musical features for genre classification. Additionally, we used confusion matrices to analyze where our models tended to perform well and poorly in terms of genres.

Results

Experiment 1:

Custom Model

Between the tuned CNN and CRNN models, the CNN had a higher overall classification accuracy in the test set with 62.5% while CRNN only had a 53.5% test accuracy.

Looking deeper into Figure 1 for the types of errors the CNN model made, largest proportion of misclassifications occurred for the genres rock (misclassified mostly as country, disco, blues, metal), blues (misclassified mostly as country, metal and rock), reggae (misclassified mostly as country, disco, hip hop and jazz), and disco (misclassified mostly as hip hop and pop). The metal and classical genres were most accurately classified. Perhaps this is because metal and classical genres have distinct qualities in comparison to country, blues and rock, which all share guitar playing.

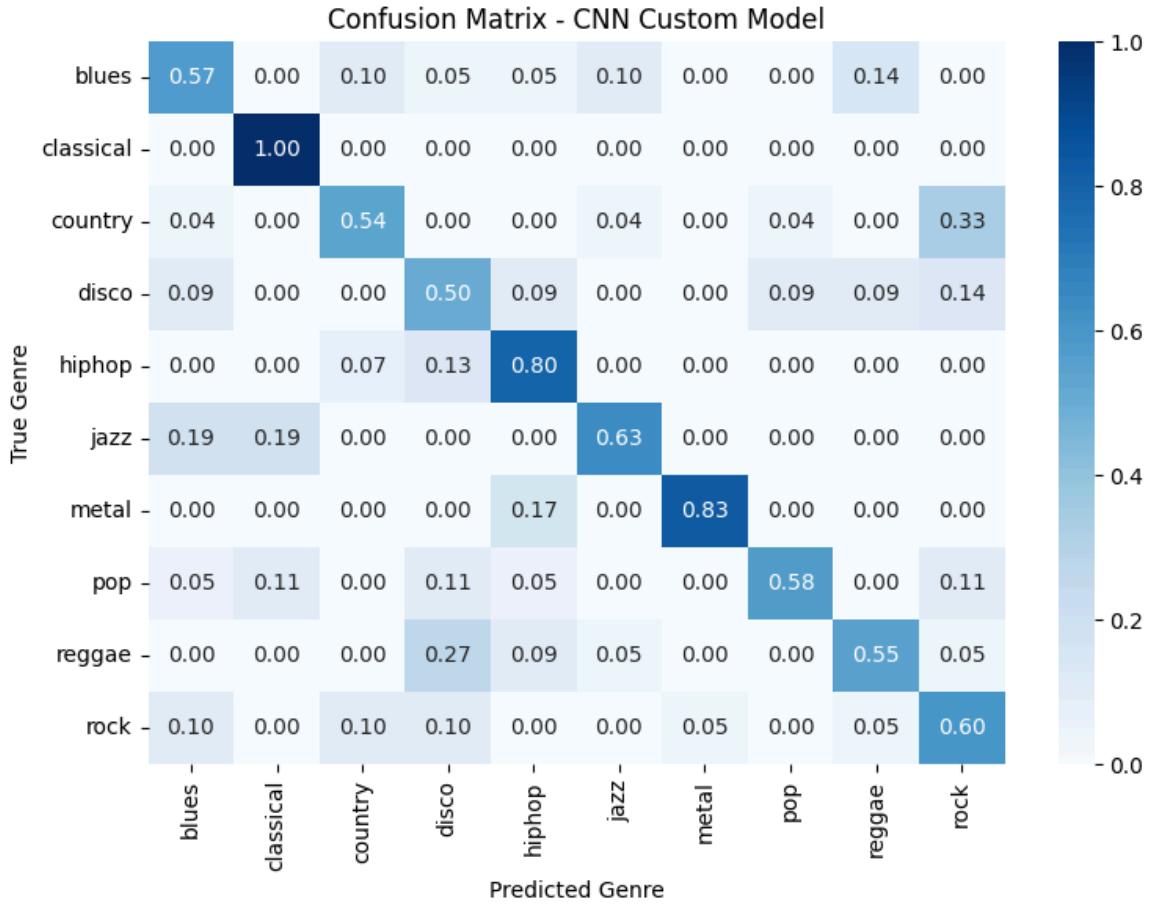

Figure 4. This confusion matrix represents the distribution of the test set's genre classification for both predicted and true labels for the CNN model. Rock, blues, reggae and disco are most disproportionately misclassified. Metal and classical are most accurately classified.

Pre-Trained Models

All of the pre-trained models showed the highest test data accuracy when all of their layers were collectively fine-tuned on the spectrogram inputs. These accuracies ranged between 59 and 76%, with VGG16 performing the best of the three models, as seen in Table 1. This performance is comparable to what Er et al. (2019) found when using spectrograms of their own dataset, AlexNet, and VGG16 to classify the mood of the audio samples, which reached a maximum of 74.2% prior to any data augmentation. The confusion matrix for the best VGG16 experimental condition is shown in Figure 2, and indicates that the model had the most trouble classifying pop (mislabeling as country or hip hop) and rock samples (occasionally confusing them for most of the other genres). The model was able to accurately identify all of the classical samples, and overall had the highest F1 scores for metal, jazz, and classical samples, which makes sense given the distinctive nature of these genres; this performance also matches what we observed in the custom model.

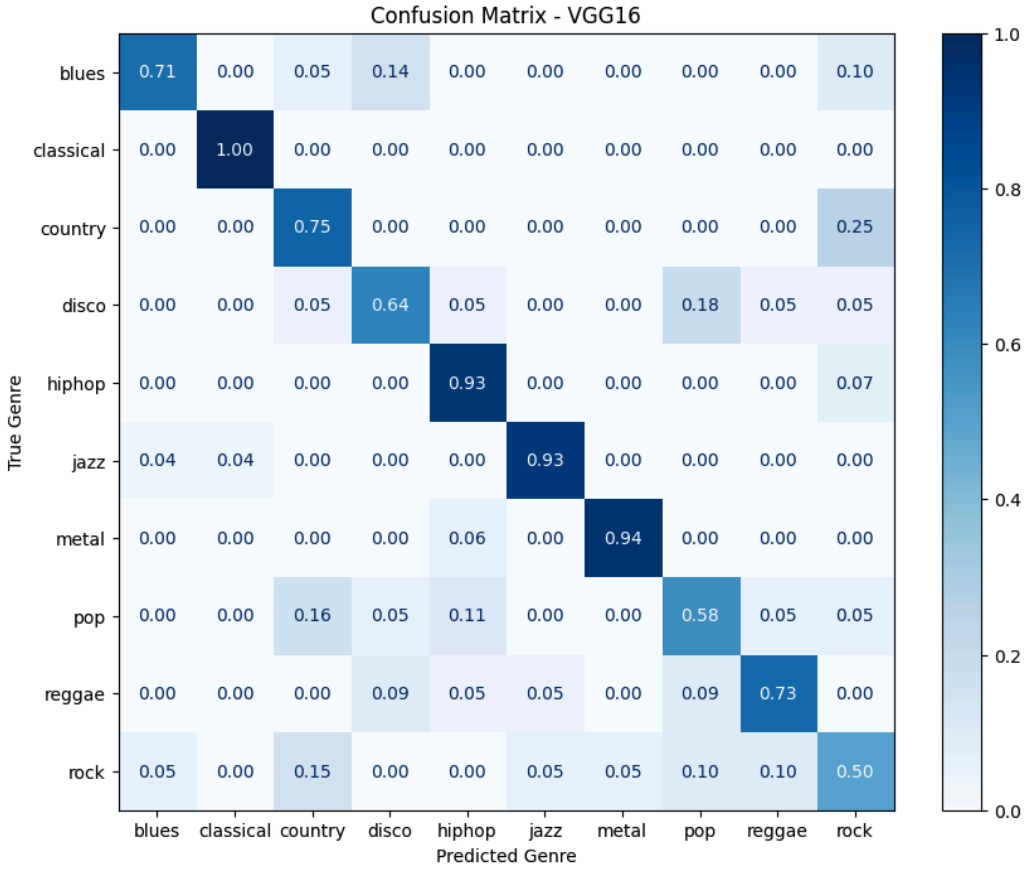

Figure 5. This confusion matrix shows on which genres the fine-tuned VGG16 model tended to perform well and poorly.

Experiment 2:

The third column in Table 1 below shows performance on the same test dataset when all of the custom and pre-trained models were trained using the augmented data. Performance improvements were considerable, with most models' accuracies increasing by around 10%. VGG16 continued to perform the best, with an accuracy of 84% after being trained on the augmented data. This increase indicates that models benefited from additional training data, as expected, and also from seeing a greater variety in the data that hopefully allowed them to generalize better to the patterns that make each genre distinct.

Table 1. Experimental conditions for each custom and pre-trained model design that yielded the best performance on the test set, along with their accuracies.

Experimental Condition	**Test Accuracy (Trained on Original Data)**	**Test Accuracy (Trained on Augmented Data)**
Custom CNN	55%	66%
Custom CRNN	45.5%	54.5%
AlexNet, Fine-Tuning Whole Model	67%	73.5%
VGG16, Fine-Tuning Whole Model	76%	84%
ResNet, Fine-Tuning Whole Model	59%	67%
Random Classifier (Reference)	10%	10%

Experiment 3:

Raw Waveform Inputs Using pre-trained Model

Raw waveform-based approaches like Wav2Vec2 are not yet as widely adopted and well-established as spectrogram-based methods. When we tried to find existing papers applying Wav2Vec2 to music genre classification, we found that there were not many. There is one study that is most relevant and comparable to ours: Kim et al. (2023) explored various parameter-efficient tuning methods applied to both spectrogram and raw waveform inputs using pre-trained models, including Wav2Vec2. In their music genre classification task on the GTZAN dataset, they compared techniques such as linear probing, input prompting, embedding prompting, adapter modules, and a combined method called Integrated Parameter-Efficient Tuning (IPET). Among these, Wav2Vec2 with linear probing achieved the lowest test accuracy of 67.8%, and IPET achieved the highest test accuracy of 78.7%.

In our experiments, we evaluated two implementations of Wav2Vec2-based models.
The first experiment achieved a moderate 61% accuracy on the test set, with notably strong performance in classifying classical music (80% recall) and metal (80% recall), while struggling with rock and disco genres (both at 30% recall).

Our second experiment incorporated significant improvements to the training methodology, including an adaptive learning rate scheduler, progressive unfreezing of the encoder layers, and enhanced regularization through additional layers of batch normalization. These modifications yielded substantial gains, increasing overall accuracy to 68%. The improved model again demonstrated strong performance on classical music (95% recall) and metal (90% recall), while maintaining similar challenges with the rock genre (30% recall).

The confusion matrix revealed interesting patterns of misclassification, with rock frequently confused with country and blues, while pop was often misidentified as reggae. These results highlight both the promise of transfer learning from pretrained audio models for music classification tasks and the persistent challenge of distinguishing between acoustically similar genres, particularly those with overlapping instrumental and structural characteristics.

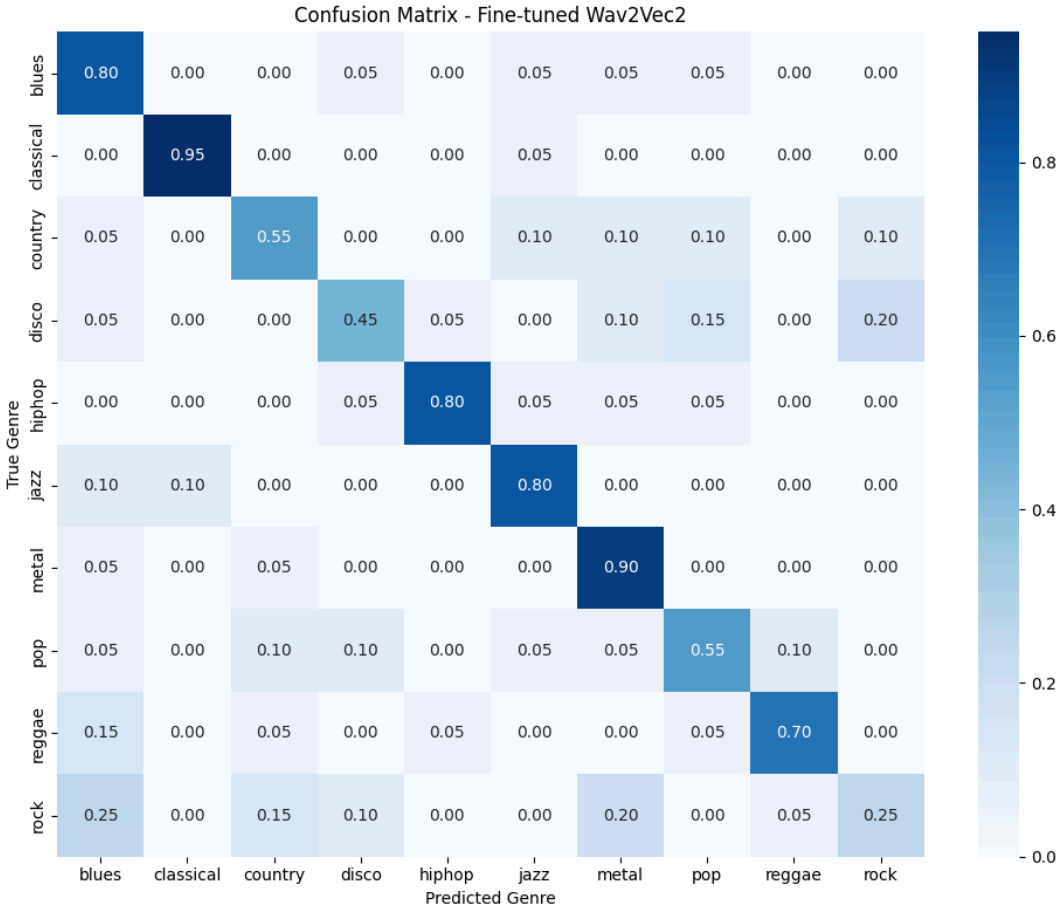

Figure 6. Confusion matrix for the fine-tuned Wav2Vec2 model.

Musical Features

Several classifiers were trained using pre-extracted musical features, and their accuracies were evaluated on the test set. The accuracies are presented in Table 3. The best performance was achieved by the tuned XGBoost model, with an accuracy of 79% (details of the parameters used to achieve this performance can be found under Appendix C), followed closely by the base XGBoost model (77.5%) and the Random Forest model (75.5%), the neural network achieved an accuracy of 75.50%. The lowest-performing model was the custom neural network built primarily with LSTM cells with 54%. The both neural networks' architectures are documented in Appendix B.

It generally classifies most genres accurately but does encounter some difficulties distinguishing between closely related styles. For instance, roughly 19% of country samples are misclassified as blues and 5% of blues as country-like because both share similar instrumentation and chord progressions. Disco tracks are confused with hip-hop and metal about 10% of the time, reflecting overlapping rhythmic or electronic elements. However, the main misclassification issues occur in the reggae and rock genres; only 52 % of reggae tracks and 67 % of rock tracks are correctly identified. To improve this, we could augment our feature set with rhythm- and timbre-based descriptors that more clearly distinguish the characteristic grooves and instrumentation of reggae versus rock.

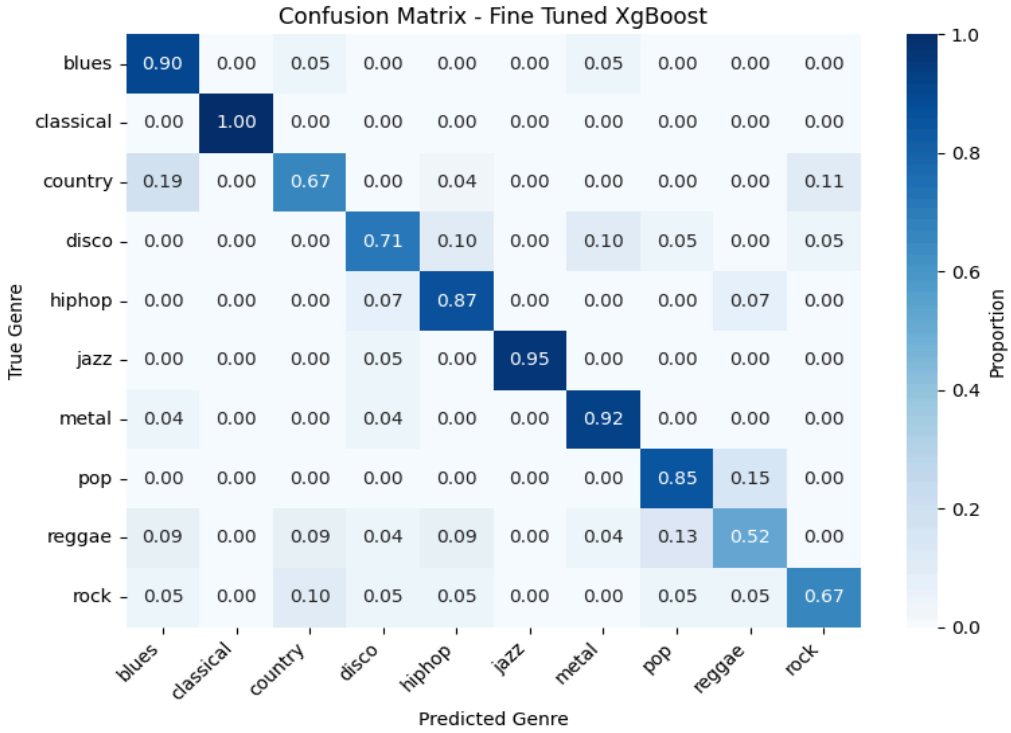

Figure 7. This figure depicts a confusion matrix for the fine-tuned XgBoost model.

Table 3. Experimental Conditions and Test Accuracies for Waveform & Musical Feature Input (Spectrogram-Based Model Performances Also Included for Comparison)

Experimental Condition	**Test Accuracy on Original Data**
Raw Waveform + Base Wav2Vec2	61%
Raw Waveform + Fine-Tuned Wav2Vec2	68%
Musical Features + KNN	64.5%
Musical Features + Random Forest	75.5%
Musical Features + XGBoost	77.5%
Musical Features + Custom NN	75.5%
Musical Features + Custom NN(LSTM)	54%
Musical Features + XGBoost (Tuned)	79%
Spectrogram + Custom CNN	55%
Spectrogram + Custom CRNN	45.5%
Spectrogram + AlexNet, Fine-Tuning Whole Model	67%
Spectrogram + VGG16, Fine-Tuning Whole Model	76%
Spectrogram + ResNet, Fine-Tuning Whole Model	59%

Experiment 4

According to Table 3, the fine-tuned XGBoost model trained on pre-extracted musical features achieved the highest accuracy on the test set, out of all models trained on the original (non-augmented) samples. This XGBoost model was subjected to two analyses. The first (Figure 7) retrains the model on 100 bootstrap samples drawn with replacement from the original training set. For Accuracy, the median is approximately 0.735, with most runs falling between 0.72 and 0.745 and only a few outliers near 0.68 or 0.78. F1 score follows a nearly identical pattern, with a median just under 0.74 and tightly clustered scores. Together, these results confirm that the original training data provides the most stable and optimal performance.

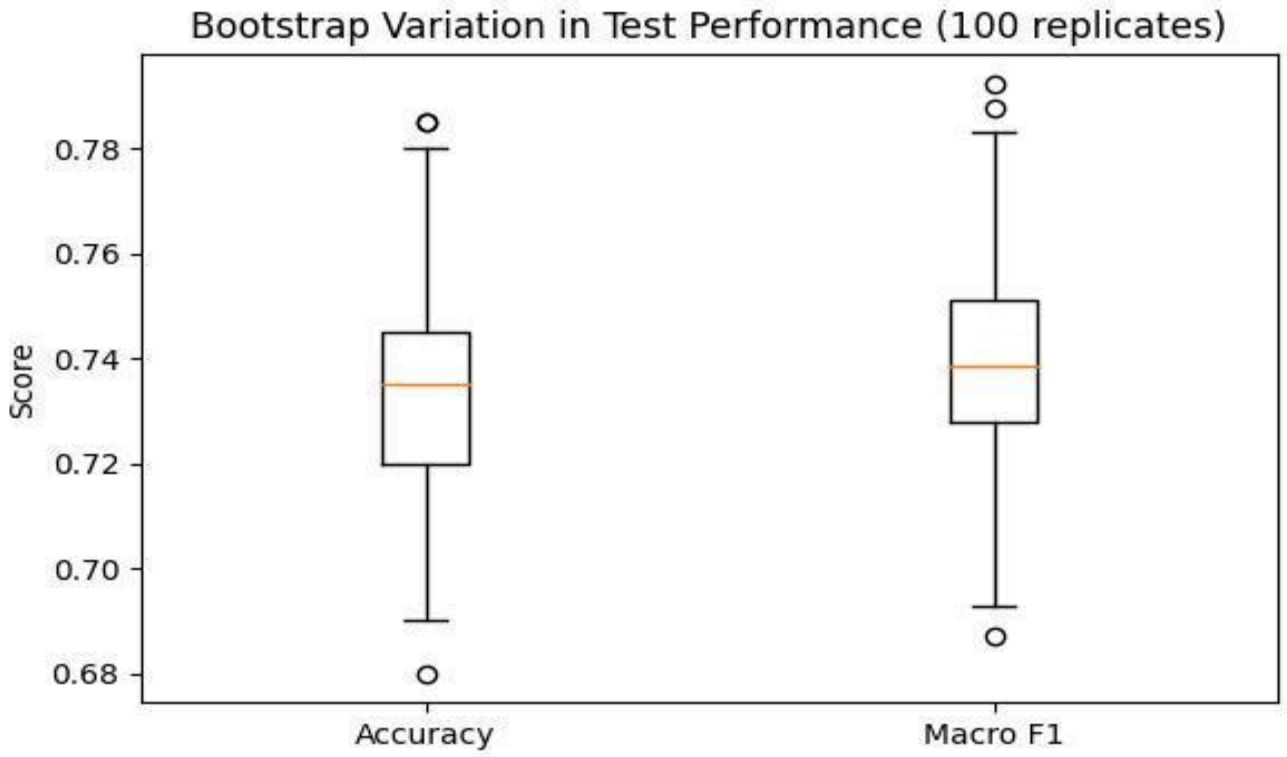

Figure 8. This box plot illustrates the variability in accuracy and F1 scores across bootstrapped samples of the fine-tuned XGBoost model. Each box spans the interquartile range (middle 50% of scores), the central line marks the median, and the whiskers (with individual points) show the full range and any outliers.

Second, we determined the point at which adding more data yields minimal gains when accuracy plateaus. To do this, we retrained the model on randomly sampled subsets of increasing size and tracked test accuracy. The model reaches near-optimal performance with 80% of the training data and achieves its highest accuracy at 90%. The slight dip at 100% likely reflects the inclusion of additional noisy samples. Overall, using more than roughly 80% of the data delivers virtually the same performance.

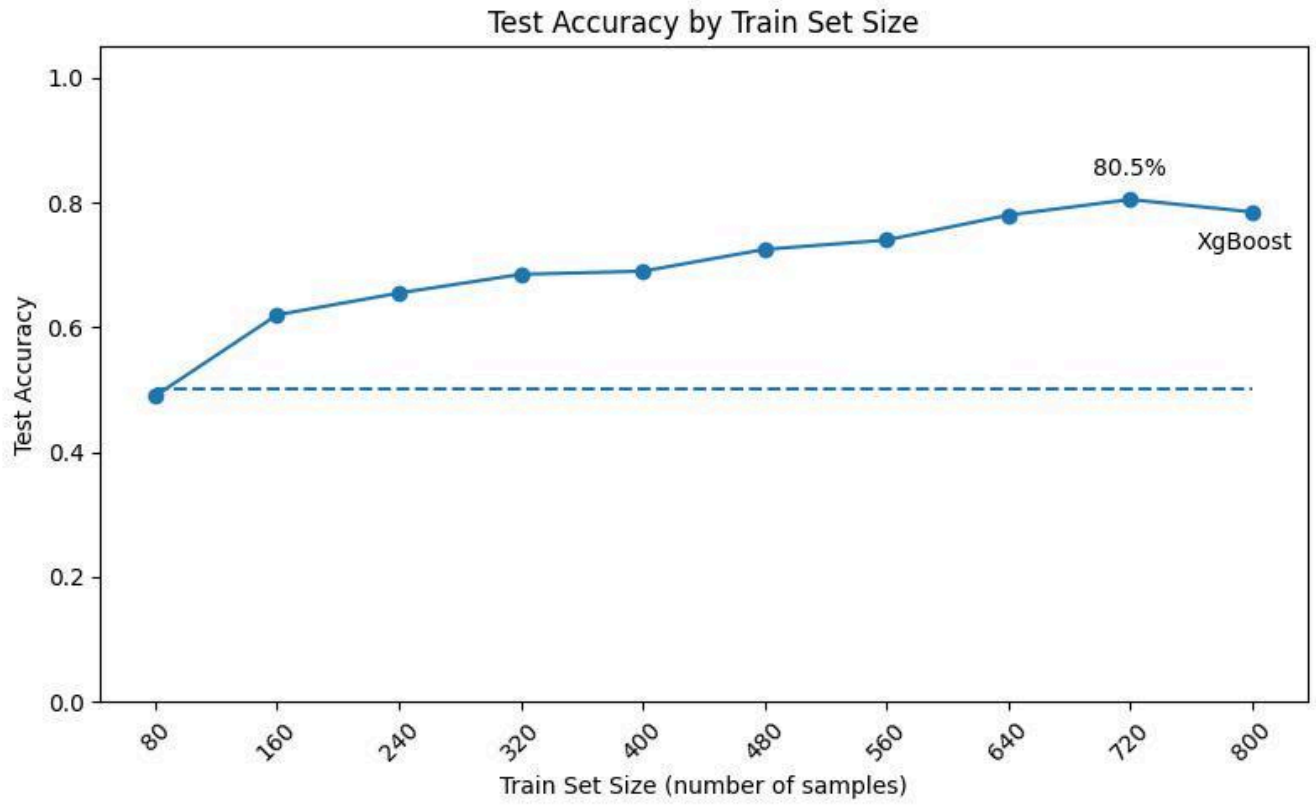

Figure 9. This line chart traces how test accuracy grows as you feed the fine-tuned XgBoost model with more training examples.

Ethical Considerations

One ethical consideration for our project is that our training samples are biased towards specific genres. The audio samples are evenly distributed among 10 different genres that are common in the U.S.; however, there are a number of international and other genres which our models will not be trained on, and thus our models will not be able to recognize and classify music from these other genres, meaning their usefulness is limited to mostly Western countries. In addition, while we did not have metadata on the artists or producers of the audio files in our dataset, it is possible that popular artists are overrepresented in our dataset, meaning sampling bias could be at play. How well our models perform on audio from lesser known artists remains an important avenue of future study to ensure that all artists' works can be classified equally well in terms of genres.

Another ethical aspect for us to consider is related to user privacy. While we are not using any real user data to train our models, in the future if a software tool were built using our models for automatic music genre classification, it is possible that other personal information would be collected in order to provide users music recommendations, in which case user data would need to be stored securely. Model interpretability would also become more important in this case as it would allow for users to more clearly see why certain songs are being recommended to them (but for the scope of this project it is not a very high priority).

Finally, we also need to consider the fact that the source of the dataset, Tzanetakis & Cook's 2002 paper, does not detail if there are any copyright concerns around the samples that they collected. While the data is publicly available on Kaggle for research purposes, if our models were to be used in any other way beyond this project, we would need to contact the original researchers to ensure that we respect copyright laws appropriately.

Conclusion

Our comprehensive investigation into music genre classification compared three input representations (spectrograms, raw waveforms, and extracted musical features) across various model architectures. For spectrograms, we evaluated both custom CNNs/CRNNs and fine-tuned pre-trained models (AlexNet, VGG16, ResNet). For raw waveforms, we implemented the Wav2Vec2 approach. For musical features, we compared traditional machine learning algorithms (KNN, Random Forest, XGBoost) with neural network approaches (custom NN, LSTM). We also investigated the relationship between training data size and model performance.

The results demonstrated that models trained on pre-extracted musical features outperformed both spectrogram-based and raw waveform approaches. The feature-based fine-tuned XgBoost model achieved the highest overall accuracy (79%), compared to 76% for the best spectrogram model (fine-tuned VGG16) and 68% for the raw waveform approach (fine-tuned Wav2Vec2). However, when data augmentation was applied to the spectrogram models, VGG16 reached 84% accuracy, showing significant improvement. This demonstrates that for visual representations of audio, augmentation techniques like time stretching and shifting can substantially enhance model performance, suggesting that such approaches should be considered standard practice for spectrogram-based classification systems. Additionally, our train split experiments revealed that more data generally improves performance.

Despite the popularity of end-to-end deep learning models, our results indicate that carefully extracted musical features still provide valuable information that can be leveraged effectively by even relatively simple neural network architectures. The strong performance of the feature-based models may be attributed to their ability to capture domain-specific musical characteristics that are directly relevant to genre classification, whereas the spectrogram and waveform models must learn these relationships from scratch with limited training data.

Our experiments also revealed interesting patterns in misclassification across different models. All approaches demonstrated high accuracy in classifying classical and metal genres, but they also struggled with genres like rock and pop. These recurring challenges suggest that certain genre confusions may stem from inherent similarities in audio characteristics rather than model architecture limitations.

These results highlight the complex relationship between input representation and model architecture, suggesting that successful music classification systems require thoughtful alignment between the two rather than simply applying the latest deep learning techniques. Future work could explore hybrid approaches that combine the strengths of both feature-based and raw data methods, as well as further investigating the impact of data augmentation techniques to address the limitations of small datasets in music genre classification.

Roles

- Javidan: Responsible for musical feature-based classification model training, experiment 4, and corresponding sections in experiment designs and results.
- Jennifer: Responsible for pre-trained models for waveform input, and its corresponding sections in experiments and results, as well as abstract, introduction, background, conclusion sections of report.
- Jahnavi: Responsible for spectrogram preprocessing, custom model training using spectrogram input, and corresponding sections in experiments and results, as well as supporting pre-trained model fine-tuning for waveform input.
- Rishika: Responsible for pre-trained model fine tuning for spectrogram input (AlexNet, VGG16, VGGish) and corresponding sections in experiments and results, as well as initial outlining of experiments and ethical considerations sections.

Appendix

Appendix A.

This table details the different fine tuning approaches that were tried with the pre-trained deep neural networks for spectrogram-based classification. The approaches that resulted in the best test set accuracy for each model are highlighted in yellow.

Model	**Approach**	**Test Accuracy**
AlexNet	Fine-tune and use first fully connected layer (layer 6) as final classification layer (output size = 10)	57%
	Fine-tune and use second fully connected layer (layer 7) as final classification layer	59.5%
	Fin- tune and use third fully connected layer (layer 8) as final classification layer	58.5%
	Unfreeze and fine-tune all layers' weights (convolution and fully connected)	67%
VGG16	Fine-tune and use first fully connected layer as final classification layer (output size = 10)	62%
	Fine-tune and use second fully connected layer as final classification layer	62%
	Fine-tune and use third fully connected layer as final classification layer	63.5%
	Unfreeze and fine-tune all layers' weights (convolution and fully connected)	76%
ResNet	Fine- tune only last layer for some epochs, then unfreeze and fine-tune all weights	59%

Appendix B:

The custom neural network model begins with two large Dense→Dropout→BatchNorm blocks of 512 neurons each, giving it plenty of capacity to learn complex feature relationships. It then steps down through four smaller Dense→Dropout→BatchNorm stages (256, 128, 64, and 16 neurons), which helps refine the signal and guard against overfitting. A final Dense layer with 10 neurons produces one score for each music genre. Altogether, the network has about 207 thousand parameters, mostly trainable weights and biases, plus a small set of BatchNorm statistics. The architecture of the model illustrated below.

Layer (type)	Output Shape	Param #
dense_48 (Dense)	(None, 512)	30,208
dropout_39 (Dropout)	(None, 512)	0
batch_normalization_39 (BatchNormalization)	(None, 512)	2,048
dense_49 (Dense)	(None, 256)	131,328
dropout_40 (Dropout)	(None, 256)	0
batch_normalization_40 (BatchNormalization)	(None, 256)	1,024
dense_50 (Dense)	(None, 128)	32,896
dropout_41 (Dropout)	(None, 128)	0
batch_normalization_41 (BatchNormalization)	(None, 128)	512
dense_51 (Dense)	(None, 64)	8,256
dropout_42 (Dropout)	(None, 64)	0
batch_normalization_42 (BatchNormalization)	(None, 64)	256
dense_52 (Dense)	(None, 16)	1,040
dropout_43 (Dropout)	(None, 16)	0
batch_normalization_43 (BatchNormalization)	(None, 16)	64
dense_53 (Dense)	(None, 10)	170

Total params: 207,802 (811.73 KB)
Trainable params: 205,850 (804.10 KB)
Non-trainable params: 1,952 (7.62 KB)

The last custom model was trained with an LSTM layer of 256 units, whose internal memory cells and gating mechanisms learn which temporal patterns to remember or forget across the audio sequence. A Dropout layer then randomly silenced some of those 256 outputs to guard against

overfitting. Next, two Dense→Dropout blocks (128 units then 64 units) progressively condensed and re-regularized the LSTM's sequential representation. Finally, a Dense layer with 10 neurons produced one score per genre, and with about 306, 000 total parameters, this setup leverages the LSTM cell's capacity to capture long-range dependencies in the audio features. The architecture of the model depicted below.

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 256)	264,192
dropout (Dropout)	(None, 256)	0
dense (Dense)	(None, 128)	32,896
dropout_1 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 64)	8,256
dropout_2 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 10)	650

Total params: 305,994 (1.17 MB)
Trainable params: 305,994 (1.17 MB)
Non-trainable params: 0 (0.00 B)

Appendix C:

The following hyperparameter set was found to be optimal for our fine-tuned XGBoost model in this scenario:


```
{
   "subsample": 0.6,
   "reg_lambda": 1.5,
   "reg_alpha": 0.1,
   "n_estimators": 1000,
   "max_depth": 9,
   "learning_rate": 0.05,
   "gamma": 0,
   "colsample_bytree": 0.6
}
```

References

Akella, R. (2019). *Music mood classification using convolutional neural networks* (Master's project, San Jose State University). SJSU ScholarWorks. https://doi.org/10.31979/etd.6cnh-j963

Baevski, A., Zhou, H., Mohamed, A., & Auli, M. (2020). *wav2vec 2.0: A framework for self-supervised learning of speech representations*. arXiv preprint arXiv:2006.11477.

Chang, P.-C., Chen, Y.-S., & Lee, C.-H. (2021). *MS-SincResNet: Joint learning of 1D and 2D kernels using multi-scale SincNet and ResNet for music genre classification* [Preprint]. arXiv. https://arxiv.org/abs/2109.08910

Costa, Y. M. G., Oliveira, L. S., Koerich, A. L., & Gouyon, F. (2011, July). *Music genre recognition using spectrograms*. Proceedings of the International Conference on Systems, Signals and Image Processing (IWSSIP). https://www.researchgate.net/publication/224251903

Er, M. B., & Aydilek, I. B. (2019). *Music emotion recognition by using chroma spectrogram and deep visual features*. International Journal of Computational Intelligence Systems, 12(2), 1622–1634. https://doi.org/10.2991/ijcis.d.191216.001

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

Kim, J.-H., Heo, J., Shin, H.-S., Lim, C.-Y., & Yu, H.-J. (2023). *Integrated parameter-efficient tuning for general-purpose audio models*. arXiv preprint arXiv:2211.02227. https://doi.org/10.48550/arXiv.2211.02227

Kurama, V. (2024, September 25). *A review of Popular Deep Learning Architectures: Alexnet, VGG16, and GoogleNet*. DigitalOcean. https://www.digitalocean.com/community/tutorials/popular-deep-learning-architectures-alexnet-vgg-googlenet

M. Ahmed et al., "Musical Genre Classification Using Advanced Audio Analysis and Deep Learning Techniques" in IEEE Open Journal of the Computer Society, vol. 5, no. 01, pp. 457-467, null 2024, doi: 10.1109/OJCS.2024.3431229.

Meng, Y. (2024). *Music genre classification: A comparative analysis of CNN and XGBoost approaches with Mel-frequency cepstral coefficients and Mel spectrograms* [Preprint]. arXiv. https://arxiv.org/abs/2401.04737

Padial, J., & Goel, A. (2011). *Music mood classification* (CS229 Final Project Report, Stanford University). Retrieved from https://cs229.stanford.edu/proj2011/GoelPadial-MusicMoodClassification.pdf

Patil, S. A., Pradeepini, G., & Komati, T. R. (2023). *Novel mathematical model for the classification of music and rhythmic genre using deep neural network*. Journal of Big Data, 10(108). https://doi.org/10.1186/s40537-023-00789-2

Pooja, M. Sri. *Steps to Convert Audio Clip to Spectrogram*. Kaggle, https://www.kaggle.com/code/msripooja/steps-to-convert-audio-clip-to-spectrogram. Accessed 21 Mar. 2025.

Residual networks (resnet) - deep learning. GeeksforGeeks. (2025, April 7). https://www.geeksforgeeks.org/residual-networks-resnet-deep-learning/

Shon, S., Tang, H., & Glass, J. (2021). Self-supervised learning for speech emotion recognition: A comparative study. INTERSPEECH.

Thome, Carl. *GTZAN Genre Collection*. Kaggle, https://www.kaggle.com/datasets/carlthome/gtzan-genre-collection. Accessed 21 Mar. 2025.

Wang, T., Li, H., Zhang, Y., & Lian, J. (2022). Exploring wav2vec 2.0 for music genre classification. In ICASSP 2022.

Zhang, Y., & Li, T. (2025). *Music genre classification with parallel convolutional neural networks and capuchin search algorithm*. Scientific Reports. https://doi.org/10.1038/s41598-025-90619-7