Music Genre Classification via CNNs using the GTZAN dataset

Problem Statement

Music streaming services have become increasingly popular in the past decade, with an estimated 78% of people listening to music via some sort of streaming service as of 2023 [1]. As the volume of digital music continuously grows larger and larger, being able to effectively classify music into genres has become increasingly important for enhancing user experiences in various applications. This can facilitate music recommendation systems, discovery, and enhance the organization of musical libraries.

Consider a service such as Spotify, which relies on classification to provide personalized playlists to its users. The accurate identification of a track's genre facilitates the continued suggestion of similar songs, understanding user preferences, and increasing customer retention. In particular, neural networks play a key part in the classification of genres as they lead to a more tailored experience and enhance the performance of recommendation algorithms [2].

In order to approach the challenge of classifying musical genres, we will utilize the Convolutional Neural Network (CNN). This implementation of neural networks is popular in image processing can also learn and extract intricate features from audio signals, and can significantly improve our chances of accurate classification of music. We will process a comprehensive feature set provided by the GTZAN data set to capture nuanced patterns within the music. This deep learning approach should allow for us to streamline our classification process and adapt to a variety of different genres.

Now that we understand our problem: the accurate classification of musical genres, and our general approach: CNNs, we will discuss our core data source used to train our model to approach this problem.

Data Source

The GTZAN dataset is a widely-used public dataset for the implementation of machine learning in music genre recognition. Collected in 2001 from a variety of sources including radio and compact discs, the GTZAN dataset is comprised of two directories and two CSV files:

- **genres_original**: 1000 30-second audio files across 10 different genres: blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae, and rock
- **images_original**: Mel spectrograms of each audio file to serve as visual representation
- **features_30_sec**: the mean and variance for multiple features for each audio file
- **features_3_sec**: the mean and variance for multiple features of each audio file, which were split into 3-second segments

The features for which the mean and variance are computed and compiled in each CSV file are described in Table 1. While it is a widely used collection of audio tracks, the GTZAN dataset does have some challenges that make classification difficult. Chief among these is the overlap of genres for particular tracks that can confuse machine learning models (in our case, a convolutional neural network) and lead to imperfect results and lower accuracy. This is especially apparent in cases of overfitting, as models may perform drastically worse on testing data if overfit to outliers and genre overlap in the training set.

Feature	Description	What does it identify?
chroma_stft	(chroma short-time fourier transform) 12 different pitch classes	which notes are present (A, B, C, etc.)
rms	root mean square of the amplitude	loudness and quietness
spectral_centroid	center of mass of the frequency spectrum	brightness and darkness
spectral_bandwidth	range of frequencies	single notes vs. noise
rolloff	frequency below which most of the signal's energy is contained	bass-heavy vs. treble-dominant
zero_crossing_rate	rate at which the signal crosses zero amplitude	percussive and noisy elements
harmony	harmonic content and pitched elements	harmonic structures like chords, melodies
perceptr	percussive content and transient elements	percussion and other rhythmic elements
tempo	beats per minute (BPM)	speed
mfcc 1-20	Mel-frequency cepstral coefficients	timbre (tone color)

Table 1: Some key features of the GTZAN data set with a brief explanation of what this feature helps identify. See the "Background" section for a more in-depth explanation of some of this terminology.

Background

In the study of music for machine learning, several fundamental elements contribute to the overall experience and perception of sound, as well as the underlying data for analysis. The following list outlines key concepts essential for understanding musical composition and performance. Each term describes a specific characteristic of music, providing insight into how various components can be quantified and processed by algorithms. These elements include pitch, melody, harmony, rhythm, tempo, and timbre, each playing a crucial role in shaping the sound and informing machine learning models aimed at tasks such as music classification, generation, and recommendation.

- **Pitch** describes the quality of a sound that makes it higher or lower. For example, a piano will produce a low pitch if a key on the left side is played, and will produce a higher pitch if a key on the right side is played. A higher frequency generates a higher pitch.
- **Melody** describes an organized sequence of notes that are perceived as a single musical concept. It's like the tune or main theme of a song that might get stuck in your head (if it's catchy enough!).
- **Harmony** describes the occurrence of multiple notes being played at the same time to create a richer sound and add depth to the music. For example, in a choir, the singers who use different vocal pitches (like tenor singers and bass singers) create the overall effect of harmony when singing at once.
- **Rhythm** describes the repeating and often predictable pattern of beats and silences in music, which creates the "pulse" of the song. It's what makes you want to tap your foot or dance to the music.
- **Tempo** describes the speed at which the music is played. From rhythm, we observe the number of beats per minute (BPM) and use this measurement to describe the overall speed.
- **Timbre** describes the "color" or quality of the sound despite the note being played. For example, a trumpet has a bright timbre, while a cello has a warm timbre.

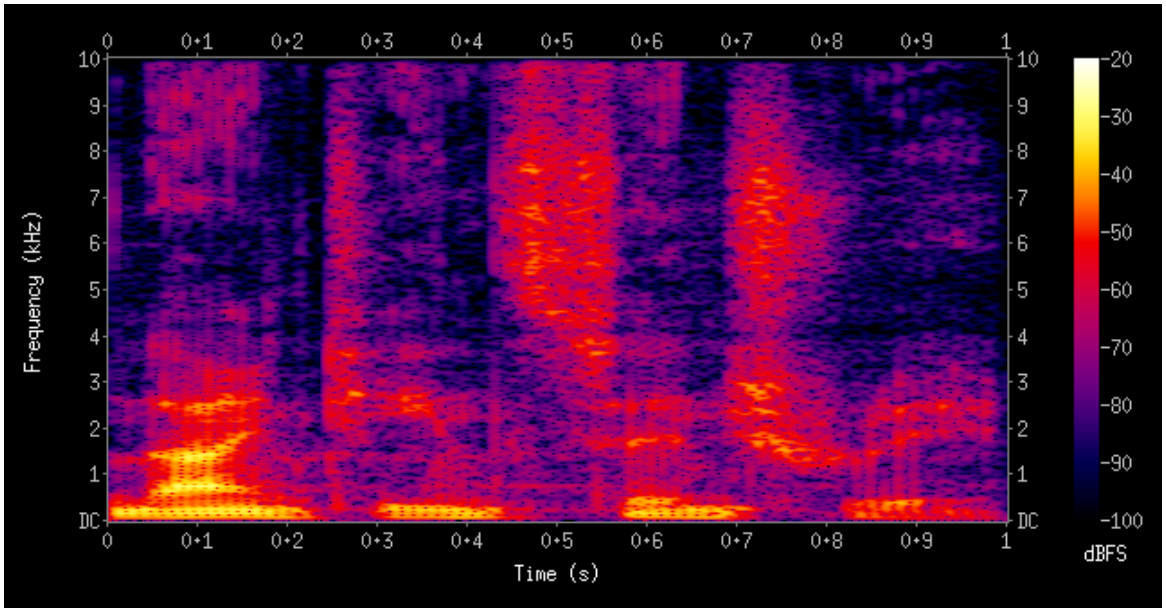

Figure 1: Spectrogram of the spoken words "nineteenth century."

Spectrograms

A spectrogram is a visual representation of a spectrum of frequencies (see **Figure 1**). One axis represents time, a second represents the frequency range, and a third shown by intensity of color represents the amplitude (or volume) of the frequency at the specific time. A spectrogram can be digitally created using a Fourier transformation. First, the data is digitally stored in a usable format and broken up in the time domain into chunks. Next, it is Fourier transformed to calculate the magnitude of the frequency spectrum for each chunk. By combining the output from each chunk at a given time, the spectrogram image is formed. This process corresponds to computing the squared magnitude of the short-time Fourier transform of the data. Using the reverse process, the spectrogram can then be reconverted into audio.

Spectrograms are useful for comprehension because they are a visual representation of how the cochlea of the inner ear encodes audio [3]. Additionally, different voices, instruments, and rhythmic patterns can all be detected through spectrogram analysis, which makes it useful for data modeling. Two distinct spectrogram structures that exist are Mel spectrogram and MFCCs, both of which will be considered in our later section on **Training and Evaluation** of the model. Both are reliant on the Mel scale, which is a scale of pitches measuring frequency bands spaced to approximate human hearing. Both MFCCs and Mel spectrograms use the Mel scale and have broad implications in the fields of speech recognition and musical recognition. **Figure 2** provides a visualization of some of the Mel spectrograms from the GTZAN data set to help us better understand the time-frequency characteristics across a small sample of different genres.

The Mel spectrogram measures a Mel scale frequency against time to display the evolution of sound. It is typically represented as a 2-dimensional image as seen in the example of **Figure 2**, where frequency bins are spaced in such a way as to approximate human hearing, *i.e.*, many bins at low frequencies and fewer bins at high frequencies. MFCCs are a derivation of the Mel spectrogram meant to extract key features of a particular signal's spectral content. The aim of this is for more efficient data compression useful in tasks such as speech recognition, music genre classification, and most every audio machine learning task. They utilize a logarithmic compression and can sometimes help reduce spectral variations not relevant to human auditory perceptions.

In training our model, we will focus on MFCCs, but will compare to Mel spectrograms to determine if better performance could be had (without increasing the model complexity significantly) by their inclusion. Both have complementary roles in representing audio data and both effectively capture the shape of a sound. Both are meant to approximate human hearing, but in different formats.

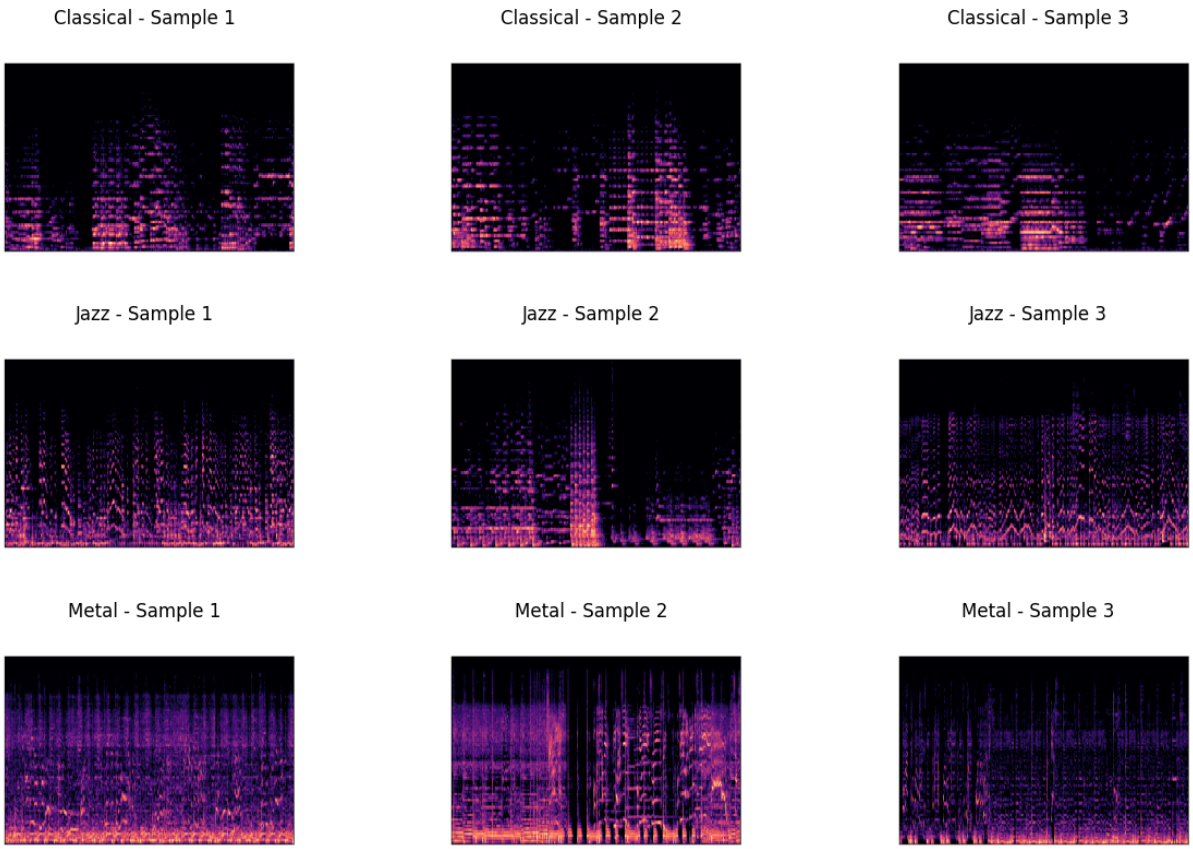

Figure 2: Mel spectrograms of a random sample of genres from the GTZAN dataset.

Methodology

Exploratory Data Analysis

To help provide a clearer view of the feature distribution within genres, we employ Kernel Density Estimation (KDE) to derive a smooth estimation of the probability density function of key features within our data (as shown in **Figure 3**). This tool is useful for exploratory data analysis and helps us understand the data's unique shape and spread but is not utilized later on in building the classification model.

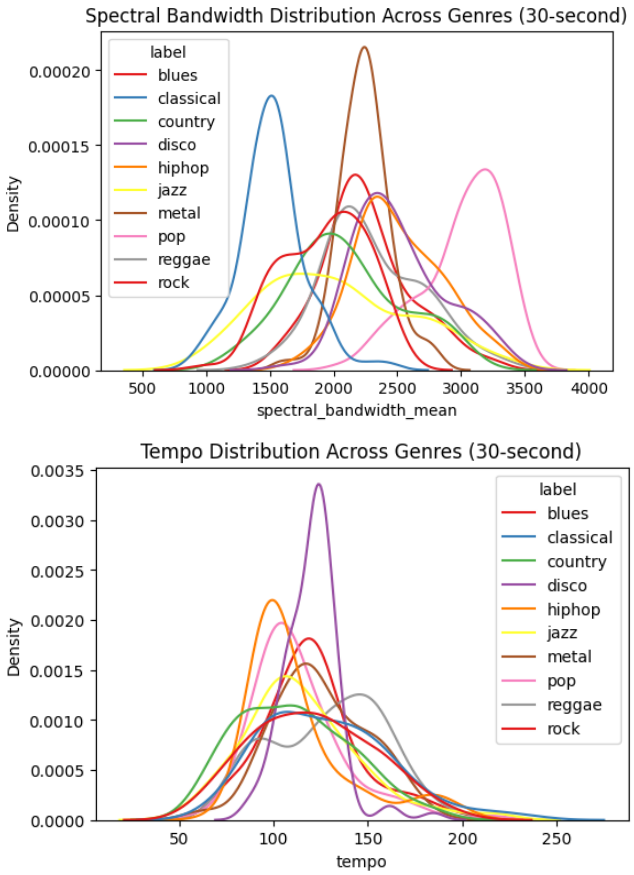

Figure 3: Kernel density estimates of certain key features, like spectral bandwidth and tempo, across all genres are shown here to help visualize how these features are distributed. From a visual inspection, we see classical, metal, and pop music have a notably different spectral bandwidth distributions (which measures how spread out the frequencies in the sound are, *i.e.*, a single note being played versus a broad range of of tones). The interpretation here is that the majority of sound in pop music is observed in single tones, while the majority of sound observed in classical covers a much broader range of tones. From observing the tempo KDE, it is also apparent that the majority of disco music is contained within a relatively fast tempo (approximately 120 BPM) without much variation. For comparison, the majority of pop songs that you would hear on the radio are distributed primarily in the 100 BPM (moderate) range.

Can this dataset be accurately represented with a fewer number of features? Feature Correlation Analysis (FCA) can be used to check for correlations between extracted audio features to see which features are strongly related, which can assist with feature selection (see **Figure 4**). Removing highly correlated features can benefit the analysis by decreasing the complexity of the model.

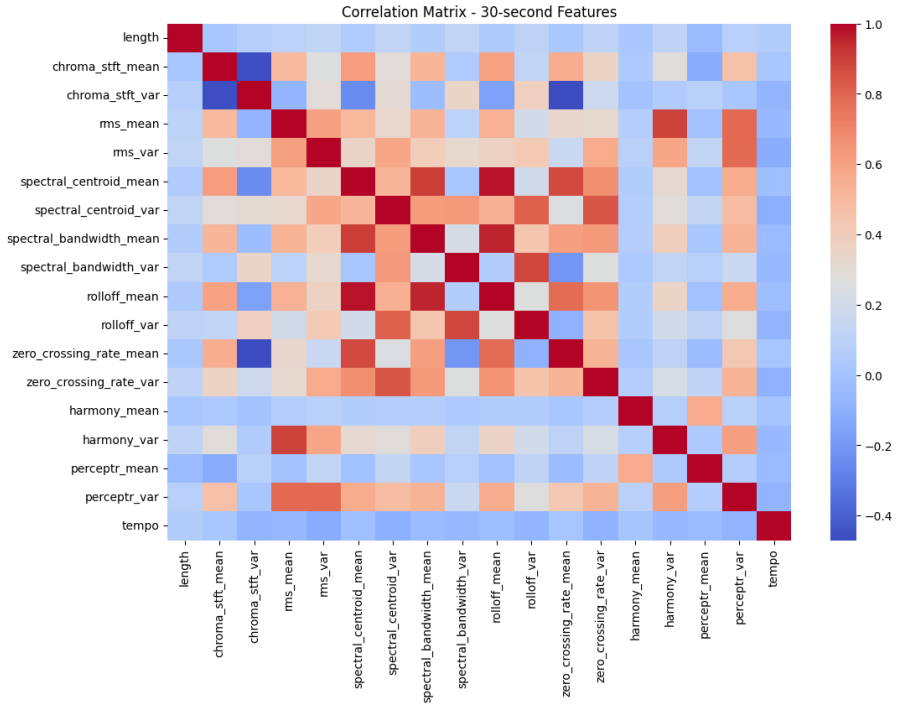

Figure 4: FCA for the numerical features of the GTZAN dataset, without the MFCC features which are inherently somewhat correlated. From a visual inspection, we can see some of the most highly correlated features are rolloff_mean and spectral_centroid_mean, as well as rolloff_mean and spectral_bandwidth_mean.

Principal Component Analysis (PCA) can also be performed to to reduce the dimensionality and visualize the genres in a lower-dimensional space. The visualization shown in **Figure 5** helps show well-separated the genres are based on the extracted features.

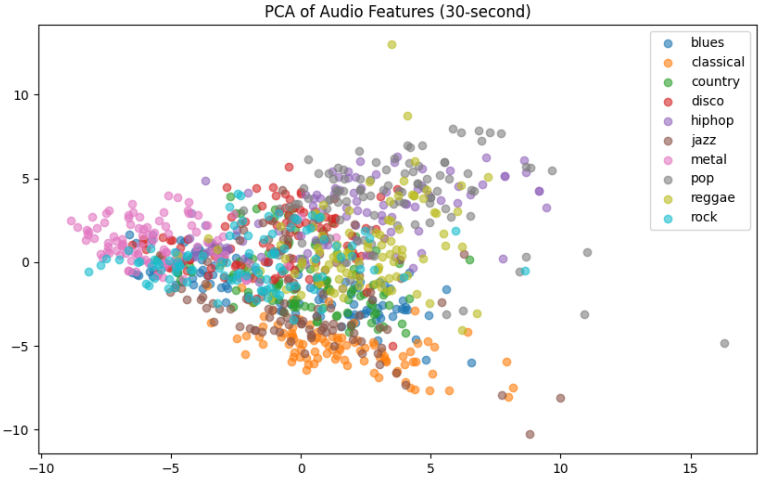

Figure 5: PCA for the 30-second audio features. The x-axis (first principal component) captures the greatest variance in the data. The y-axis captures the second greatest variance and is orthogonal to the first principal component. Genres that are closer together on the plot have similar features, while genres that are farther apart have more distinct features. From a visual inspection of the clusters, we can see that metal and rock might be similar genres, as well as classical and jazz. Hiphop and pop also seem to have some overlap.

Music Genre Classification

Model Architecture (CNN)

We will design a music genre classification model leveraging the machine learning technique, Convolutional Neural Network (CNN), to classify music genres by learning genre-specific patterns from the comprehensive feature set and spectrograms provided by the GTZAN data set. CNNs are a class of deep learning model specifically designed for processing structured grid data, and are typically reserved for only images. However, because the GTZAN dataset provides precomputed statistical features for each audio file, we're able to manipulate these features with minimal preprocessing to work with a CNN model.

CNNs mimic the way people perceive visual information, making them extremely effective in cases of classification, and with enough features of a sound file, we may accurately classify the genres of our music library. At its core, a CNN has convolutional layers, *i.e.*, layers of filters (or kernel functions) that can detect local patterns and convolve at each layer over the prior layer's output. This tends to capture statistical and visual patterns that may be present in our dataset.

Our kernel functions mentioned above, also called activation functions, can learn complex patterns in our data. An additional consideration are pooling layers; these layers will downsample data and reduce the dimensionality by selecting the maximum values from a defined window of features in our feature set, retaining the most prominent features in a similar idea to PCA. After these steps, data will be passed through the convolutional and pooling layers and be flattened. This results in a final classification that can have very interesting and useful results depending on our initialized hyperparameters.

One of the most important uses of CNNs, and one that will be leaned on in our audio recognition task, is their ability to learn patterns that are translation invariant and contain hierarchical features [4]. Translation invariance refers to the property of a model that allows it to recognize features regardless of their position in our input space. Thus an object can be identified any location in our grid. This is especially relevant to our music genre classification task as we will see repeating patterns, such as melodies and notes, in very different places even within the same genre.

These properties and uses of a CNN will be used in the training of our model and should be kept in mind. As an example, consider **Figure 6.**

Figure 6: Example of translation invariance in a Convolutional Neural Network. If a CNN has learned the image of the dog on the left, it will be able to recognize the pieces of the image in the right despite them having been translated to other parts of the input grid. Original "Dog Audio" image is by digital artist Papia Nandi.

While we are starting with only two convolutional layers and two pooling layers, we will iteratively increase the number of convolutional layers in order to determine if better results are had. We predict that as we increase the number of layers that our earlier layers will learn simple patterns (such as the edges of our MFCCs), and the deeper layers will be able to learn more complex representation that may appear, such as motifs or even possibly full musical phrases.

Similarly we will train with only two max pooling layers to start, as we are previously applying a dimensionality reduction technique prior to our CNNs usage. As mentioned in section **4.a.**, this technique is similar to PCA in that it only retains the most significant features. This should ensure our results are also not overfit as we don't fit too much of our less meaningful data in our training set.

As for the actual classification of our training data, we are using soft maximization to compute probabilities. Given that we are classifying 10 genres rather than a traditional binary classification problem, we want to train our model to output probabilities and provide a soft classification rather than a hard classification as might be seen with methods like support vector machines.

It will be beneficial then to ensure the model is fully connected with dense layers, *i.e.* that every neuron in every layer of our network is connected to every neuron of the next layer, such that we have a connected network similar to what is seen in **Figure 7**.

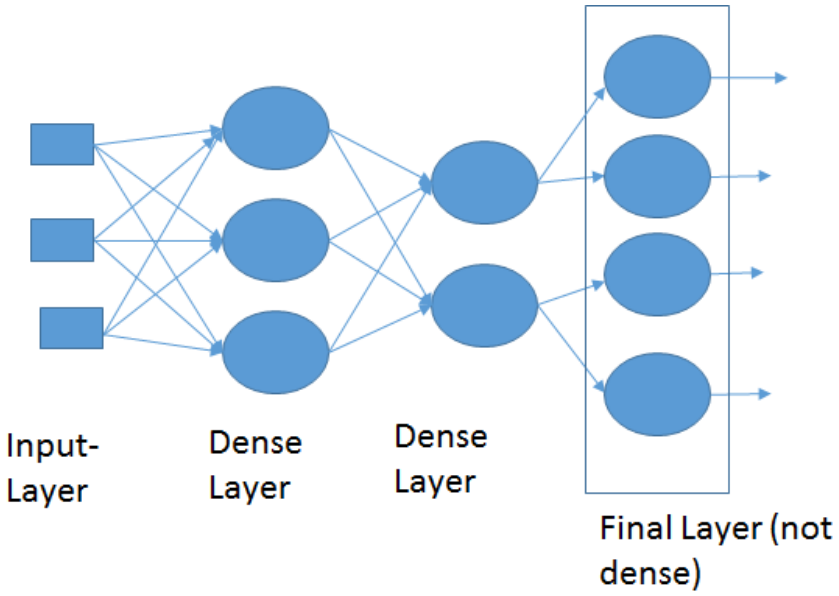

Figure 7: Example of dense and not-dense layers.

In order to capture relationships that are non-linear, as is likely in audio data, we will be using non-linear activation functions at each layer of our neural network. While we use a soft maximization strategy for our ultimate output, some of our dense layers will instead use a Rectified Linear Unit, or ReLU. This is a simple function that captures nonlinearity, creates sparsity in a network (i.e. many 0s) for computation, and avoids overfitting. It is defined as

$$ReLU(x) = max(0, x).$$

When considering all of the above, our CNN should be able to handle variations in our audio data, avoid overfitting, and be robust enough to help our model be generalized to unseen data.

Building the CNN and Feature Selection

We begin by setting a fixed random seed for Python's `random` library to ensure reproducibility across all processes in the model that involve shuffling or otherwise random initializations or sampling. We then prepare the data by defining all necessary parameters, such as the directory paths and the list of genres. A `load_image_data` function is implemented to load the Mel spectrograms from the `images_original` directory and these images are resized to (128, 128) and normalized to a range of [0, 1]. We also extract the MFCCs and precomputed statistical features (mean and variance) for each 30-second audio file from the `features_30_sec` CSV file. These features include spectral, temporal, and harmonic descriptors such as chroma short-time Fourier transform (chroma_stft), root mean square energy (RMS), spectral centroid, spectral bandwidth, rolloff, zero-crossing rate, harmony, perceptr, and tempo as described in the **Data Source** section. These will serve as representations of each track stored as multi-dimensional feature vectors with corresponding labels. We preprocess the data by converting the corresponding genre labels into numerical representations using label encoding and then into categorical format suitable for this classification task.

To help reduce multicollinearity and simplify our model, we also encode the FCA performed earlier into our model, and this time with a functional output. This brief analysis our code effectively identifies and eliminates highly correlated features (correlation coefficient ≥ 0.8), ensuring that only a single feature from each pair of highly correlated features makes it into the model. This step helps limit overall redundancy and mitigates potential confusion arising from multicollinearity, and helps create a smaller more efficient model.

We began with the full set of all statistical features, MFCCs, and spectrograms provided in the GTZAN dataset for all 1000 songs across 10 genres. We found quickly that including a combination of both MFCCs and Mel spectrograms proved did not benefit model performance, while simultaneously over-complicating the model. This can be attributed to the Mel spectrogram's inherent overlap with the MFCC, as both use a Mel scale to approximate human hearing. A Mel spectrogram may capture more raw information than an MFCC, however the patterns a CNN is likely to learn may be similar or identical in both instances, especially in the case of the simple audio data used for training. For this reason, we decided to simplify our model and remove spectrograms from the analysis entirely. The initial model required feature-based and image-based data to processed separately in a 1-dimensional and 2-dimensional CNN, respectively, so this decision simplified our model significantly and improved efficiency without sacrificing performance. Thus the network now focuses on feature extraction along the temporal or sequential structure of the audio features and does not process spatial relationships as a 2-dimensional CNN would for images.

On the remaining features, the model calculates the pairwise correlations between features and stores these in a correlation matrix. The upper triangle of the correlation matrix is selected to avoid duplicate checks, and for each feature, the code checks whether its correlation with any other feature exceeds the threshold (correlation ≥ 0.8). Only the first feature in each correlated pair is removed from the model. Features over the correlation threshold that were removed from the model were spectral_bandwidth_mean, rolloff_mean, rolloff_var, zero_crossing_rate_mean, zero_crossing_rate_var, and harmony_var, where the suffix "mean" represents the average of that feature and the suffix "var" represents the variance.

The majority of features were included in the final model, including the chroma features which emphasize tonal content. They do not inherently encode any sort of musical harmony, such as chords, but can capture underlying harmonic structures such as scales. This proves important in being able to distinguish genres like jazz which may utilize very distinct harmonic characteristics. Spectral centroids measure the "center of mass" for a particular spectrum, *i.e.* the brightness or texture of a sound. It can denote where energy is concentrated in the frequency domain and thus help to distinguish between different genres, as seen in our model. As an example, electronic music tends toward high-frequencies and to be brighter as compared to classical music which has low-frequency content more consistently. Spectral centroids add some level of robustness to our model.

It makes intuitive sense the inclusion of these would make MFCCs more robust, whereas the inclusion of Mel spectrograms produces little change. MFCCs and Mel spectrograms measure similar spectral content in audio and capture timbral features. Chroma features carry harmonic information related to tonality, chords, etc. Spectral centroids measure the brightness/timbre. RMS and tempo capture rhyhthm. Essentially, we have given our CNN the means to measure on timbre, rhythm, harmony, and recognize specific instruments. Naturally this will improve the overall quality of the model.

After correlation filtering, we standardize the remaining features to ensure consistent scaling, which is important for optimizing the CNN training. Thus our final model in being built uses a combination of MFCCs and other audio features, effectively capturing a spectrogram representation as well as other audio features that can capture spectral, temporal, timbric, and harmonic descriptions and potentially capture less easy to learn concepts such as rhythm. This will serve as the the most robust model, though testing will be done on just using MFCCs as well to see if a spectrogram representation is sufficient to predict music genres.

To improve the robustness of our evaluation, we also implement K-fold cross-validation. This helps to mitigate any biases in our performance estimates by training and validating the model across multiple subsets of our data. After data loading and preprocessing, a 5-fold cross-validation configuration is established which splits the dataset into five equal segments that will each be utilized for both training and validation.

From here we construct our sequential CNN model with a 1-dimensional architecture designed to work with time-series data. This architecture includes two convolutional layers with ReLU activation which are used to initially extract features along the time axis, followed by two max pooling layers to reduce the dimensionality of features maps and capture more meaningful patterns. We flatten our final result from a 2-dimensional output to a single dimensional vector, which is then passed through two fully connected layers. The last layer employs a soft maximization approach for activation functions in order to output probabilities each track belongs to a specific genre. Finally, we compile the model with categorical cross-entropy as the loss function and the Adam optimizer, which is a type of stochastic gradient descent method used to minimize the loss function. Because Adam is suited for adaptive learning rates, we also implement a learning rate scheduler that halves the learning rate every ten epochs. This helps to ensure convergence in a timely manner and helps the model learn finer details as training progresses.

Training and Evaluation

We use the keras package from the TensorFlow [5] platform to train the model for 50 epochs. Each fold is trained on a unique training/validation split, while sequentially recording necessary information to evaluate the performance, such as validation accuracies and true/predicted labels. The model converges in approximately 30 seconds after initialization, including data loading, preprocessing, training, and cross-validation, which is a reasonable runtime that is not excessively time-consuming nor expensive.

Ultimately, the value of a deep learning model such as a CNN comes from its ability to learn relevant features from raw data without manually engineered features. Having robust sets of features helps to learn more of these patterns. Furthermore, if the CNN is well-optimized on a generalized set of raw data, such as an MFCC, it can learn the most relevant patterns, and still produce solid predictions. Even when excluding tempo, RMS, spectral centroids, and chroma features, we still obtain an accuracy of nearly 60% when using MFCCs alone. We have included the additional features as it provides a significant boost to accuracy, but CNNs prove extremely well suited to handling this transformed audio data.

To evaluate our model, we employ classical methods for classification reporting such as the confusion matrix and precision measurements across each genre. This helps us identify not only the overall performance, but also specific areas where classification error is likely to occur. The validation accuracy scores for each fold are shown in **Table 2**. The final confusion matrix is shown in **Figure 9**. This matrix aggregates the predictions across all validation sets, providing genre-specific insights regarding which genres are correctly classified most often and which are incorrectly classified most often.

Finally, at the end of all folds, we calculate the model's average validation accuracy, which essentially summarizes the model's overall predictive performance.

Genre Confusion

As can be seen in **Figure 9**, several genres experience higher rates of confusion. These align with where one might expect these genres to become confused. For example, disco and rock have a larger portion of "genre confusion" than classical music. Similarly classical and jazz have the same type of confusions. This can partially be attributed to the nature of the training tracks: each training track is exactly 30 seconds in our GTZAN dataset. There are certain progressions, melodies, and sounds that are common to many of these genres that may not have a chance to evolve over the course of shorter tracks.

To demonstrate we will consider 3 longer form songs with distinct genres, as well as 3 with a more blended genre. All of these were randomly selected with a scrape of YouTube Music. These songs are

- **Disco.** September - Earth, Wind & Fire
- **Jazz.** Feeling Good - Nina Simone
- **Pop.** Party in the USA - Miley Cyrus
- **Pop & Rock.** Solsbury Hill - Peter Gabriel
- **Jazz & Reggae & Rock.** Rolling 40th - Himiko Kikuchi
- **Classical & Jazz.** Fire on High - Electric Light Orchestra.

Using our trained classifier to predict each of these we find the results in **Table 2**.

Song	Genre	Predicted
September	Disco	Disco
Fire on High	Classical, Jazz	Jazz
Feeling Good	Jazz	Jazz
Rolling 40th	Jazz, Reggae, Rock	Jazz
Party in the USA	Pop	Pop
Solsbury Hill	Pop, Rock	Disco

Table 2: Predictions on longer songs.

On longer songs, we see fairly good performance despite being trained on shorter tunes. This is partially due to the translation invariance discussed earlier in this paper, and its implication that a CNN will be able to recognize structures hidden in an image regardless of position. This indicates that despite our training set being composed of songs only 30 seconds long each, their applicability to longer pieces remains untarnished.

So while the length of data does not present an issue for predictions, there are other factors which cause issues within our CNN. We see genre confusion at a rate of approximately 28% for all classifications. More accurately, it would be better to point out how some genres are classified particularly well, while others experience an extreme level of confusion. Consider the example of "Solsbury Hill". This song's true label is pop rock. Our model classified it as disco. Rock and disco are our two most poorly classified genres at 55% and 54% respectively. Breaking these two out further, close to one-third of all misclassifications of rock or disco were misclassified as each other.

Then the natural question becomes "how similar are these genres?" To measure genres and determine how similar they are are via a similarity metric, we use a cosine similarity matrix, thus producing the data seen in **Figure 8**. Cosine similarity measures the angle between vectors in our data, not their magnitude. We will consider this in the context of audio analysis. Our features are normalized (in our case, scaled) to reduce the impact of amplitude on our scale. Additionally, musical features such as rhythm, melody, and timbre are more likely to be consistent between genre, cosine similarity will focus on these and ignore magnitude based factors such as amplitude. Additionally, since MFCCs exist within a high-dimensional space, similarity in direction proves more useful for comparing features than traditional distance measures such as Euclidean distance. This way we are able to focus on the overall spectral shape represented by an MFCC across frequency bands rather than specific energy values like amplitude that occur at each frequency. For measuring similarity of our genres, this is especially true as many genres may differ in spectral patterns but not in things such as volume (*e.g.*, there are loud and quiet pieces within each genre).

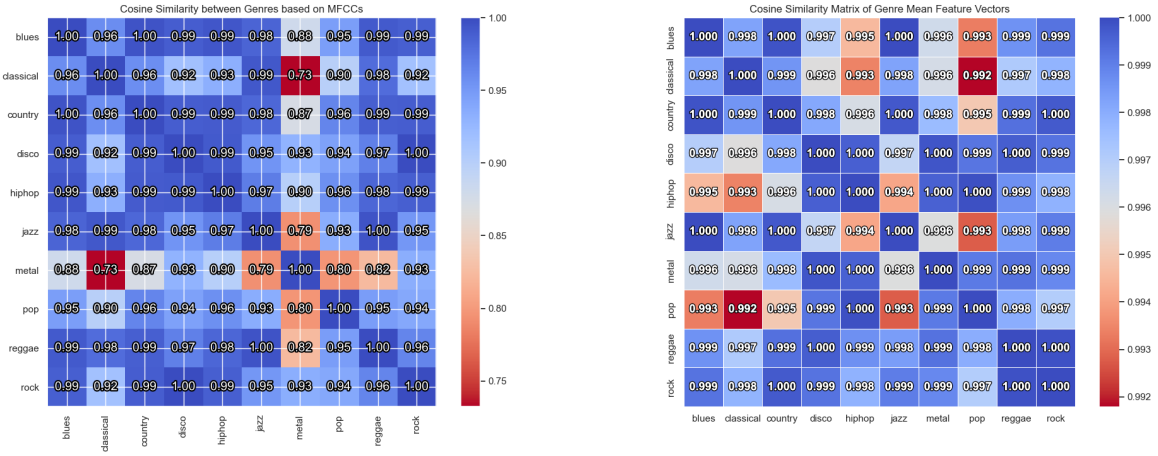

Figure 8: Cosine Similarity Scores between each genre first by MFCCs and then all features excluding MFCCs. From this figure we see that many genres are remarkably similar (typically coinciding with those we see most often misclassified). When comparing disco and rock, for example, their cosine similarity when rounded to two decimal places is practically identical. Thus we would expect our CNN to be more confused on these data points.

Examining **Figure 8**, it is more apparent why the CNN becomes confused in the places in which it does. On those genres which have highly similar MFCCs, we see the highest misclassification rate, and on those genres in which the similarity scores are less, we see the fewest number of misclassifications. In the case of metal and classical music, there was not a single misclassification of these two, as seen in **Figure 9**. This similarly holds out when examining the cosine similarity in our remaining features, though the MFCC scores more closely align with our model results. This is likely due to the number of cepstral coefficients as compared to other features.

Final Results

Fold	Validation Accuracy
n = 1	69.0%
n = 2	69.5%
n = 3	77.0%
n = 4	67.5%
n = 5	75.5%

Table 3: Validation accuracy scores for each fold.

The average validation accuracy across all five folds for this model was 0.7170, or 71.70%. Our evaluation strategy for the audio classification model endeavors to ensure performance across our ten genres while mitigating problems that may arise from feature similarity and overfitting. By utilizing classical classification metrics, K-fold cross-validation, and detailed analysis, we were able to accurately evaluate the weaknesses and strengths of this model. The iterative refinement of our feature selection and model's underlying architecture enhanced classification accuracy and has revealed deeper insights into the underlying structures and patterns inherit to audio data. These efforts will contribute to the development of an effective audio classification system, with implications and insights for various applications in music analysis, recommendation systems, and be applicable to a scope even beyond our problem statement.

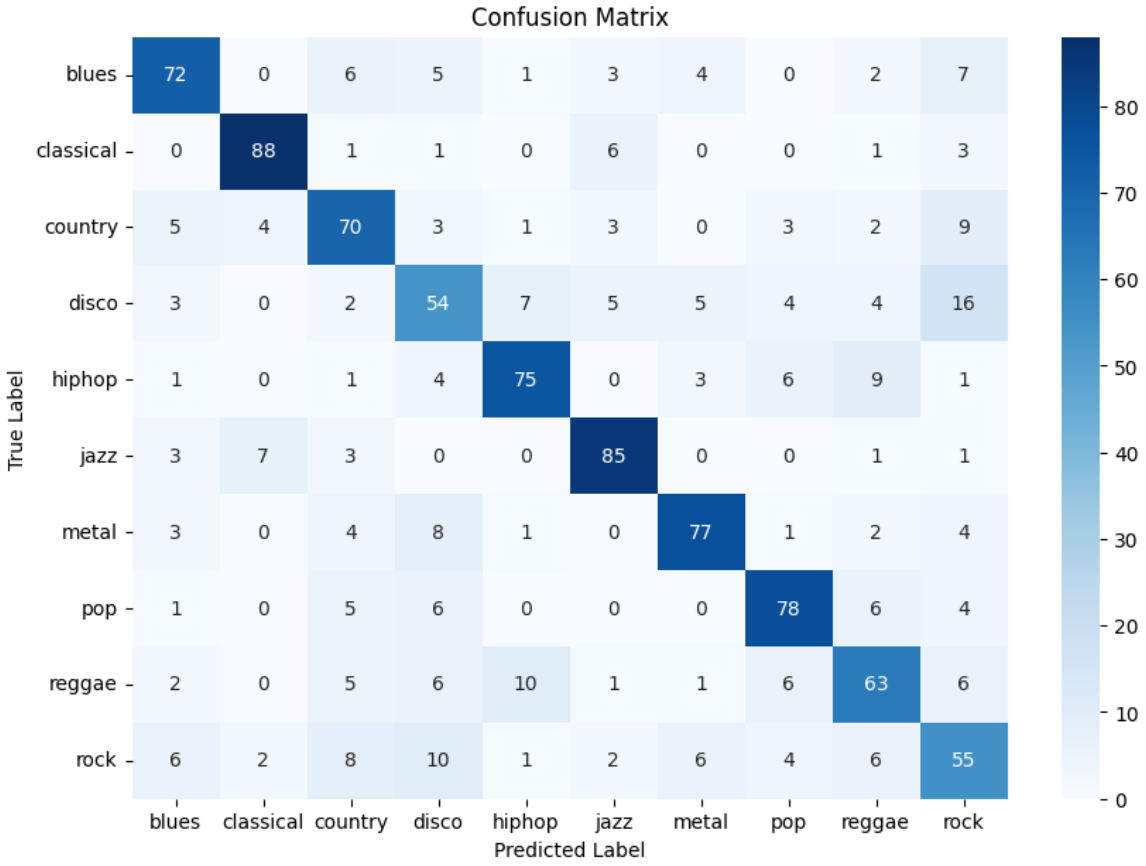

Figure 9: The resulting confusion matrix is plotted here using Seaborn's heatmap to help demonstrate the model's performance across all genres.

Conclusion

In this study, we explored the use of Convolutional Neural Networks (CNNs) for classifying music genres based on audio features extracted from the GTZAN dataset. Overall, our code integrates data preprocessing, feature selection, model training, evaluation, and visualization into a unified pipeline, providing a systematic approach to music genre classification based on statistical audio features and MFCCs. Our final model achieved an overall classification accuracy of 72%, which is a fairly decent rate but leaves room for some improvements. This result demonstrates that CNNs, with their ability to automatically learn hierarchical features from raw audio data, can be effective for music genre classification, but also highlights several challenges that must be addressed to increase performance.

To improve our model, the natural thing to do would be to increase the size of our dataset. Containing only 100 samples within the GTZAN dataset for each of our 10 classes, the dataset is fairly small. To avoid overfitting this size of dataset, several dropout layers were implemented. While improving the generalization of the model, and allowing us to achieve an accuracy score of 72%, there is room to improve. To make future improvements to such as model, collecting thousands of more samples for each class would most likely lead to better results as more patterns are found and fit by the CNN. The LibROSA library [6] in Python could be used to process additional audio files then extract the sampling rate, MFCCs, chroma features, and more from each raw audio track. These tracks and their corresponding features could be used to supplement the GTZAN data set.

The model's performance could also potentially be improved by data augmentation, such as pitch-shifting, time-stretching, and adding noise, to artificially expand the training set and expose the network to more variation. Transfer learning using pre-trained models on larger, more diverse audio datasets could help the model learn more generalizable features. Leveraging other activation functions besides ReLu or using a function other than softmax for output layers may also improve results, and further experimentation would be needed in this regard. Exponential linear units or logistic regression may be useful test functions for this model, as logistic regression performs well on multi-class data as is seen with our genre classification task.

In conclusion, this research establishes a baseline for genre classification using CNNs and serves as a stepping stone towards a more robust model. With larger datasets, better feature engineering, and more sophisticated architectures, the potential for improving genre classification performance is considerable, and future exploration should continue to explore these areas to push the boundaries of music classification with deep learning.

References

[1] Frances Moore ... Global music report 2023. *IFPI*, 2023. https://www.ifpi.org/wp-content/uploads/2020/03/Global_Music_Report_2023_State_of_the_Industry.pdf.

[2] Nikki Pelchat and Craig M Gelowitz. Neural network music genre classification. In *2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE)*, pages 1–4, 2019.

[3] J.O Smith. *Mathematics of the Discrete Fourier Transform (DFT) with Audio Applications, Second Edition.* Stanford, 2007. http://ccrma.stanford.edu/~josmdft/.

[4] F. Chollet. Deep learning with python. *New York: Manning*, 2018.

[5] TensorFlow Developers. Tensorflow, October 2024.

[6] Brian McFee, Matt McVicar, Daniel Faronbi, Iran Roman, Matan Gover, Stefan Balke, Scott Seyfarth, Ayoub Malek, Colin Raffel, Vincent Lostanlen, Benjamin van Niekirk, Dana Lee, Frank Cwitkowitz, Frank Zalkow, Oriol Nieto, Dan Ellis, Jack Mason, Kyungyun Lee, Bea Steers, Emily Halvachs, Carl Thomé, Fabian Robert-Stöter, Rachel Bittner, Ziyao Wei, Adam Weiss, Eric Battenberg, Keunwoo Choi, Ryuichi Yamamoto, CJ Carr, Alex Metsai, Stefan Sullivan, Pius Friesch, Asmitha Krishnakumar, Shunsuke Hidaka, Steve Kowalik, Fabian Keller, Dan Mazur, Alexandre Chabot-Leclerc, Curtis Hawthorne, Chandrashekhar Ramaprasad, Myungchul Keum, Juanita Gomez, Will Monroe, Viktor Andreevitch Morozov, Kian Eliasi, nullmightybofo, Paul Biberstein, N. Dorukhan Sergin, Romain Hennequin, Rimvydas Naktinis, beantowel, Taewoon Kim, Jon Petter Åsen, Joon Lim, Alex Malins, Darío Hereñú, Stef van der Struijk, Lorenz Nickel, Jackie Wu, Zhen Wang, Tim Gates, Matt Vollrath, Andy Sarroff, Xiao-Ming, Alastair Porter, Seth Kranzler, Voodoohop, Mattia Di Gangi, Helmi Jinoz, Connor Guerrero, Abduttayyeb Mazhar, toddrme2178, Zvi Baratz, Anton Kostin, Xinlu Zhuang, Cash TingHin Lo, Pavel Campr, Eric Semeniuc, Monsij Biswal, Shayenne Moura, Paul Brossier, Hojin Lee, and Waldir Pimenta. librosa/librosa: 0.10.2.post1, May 2024.