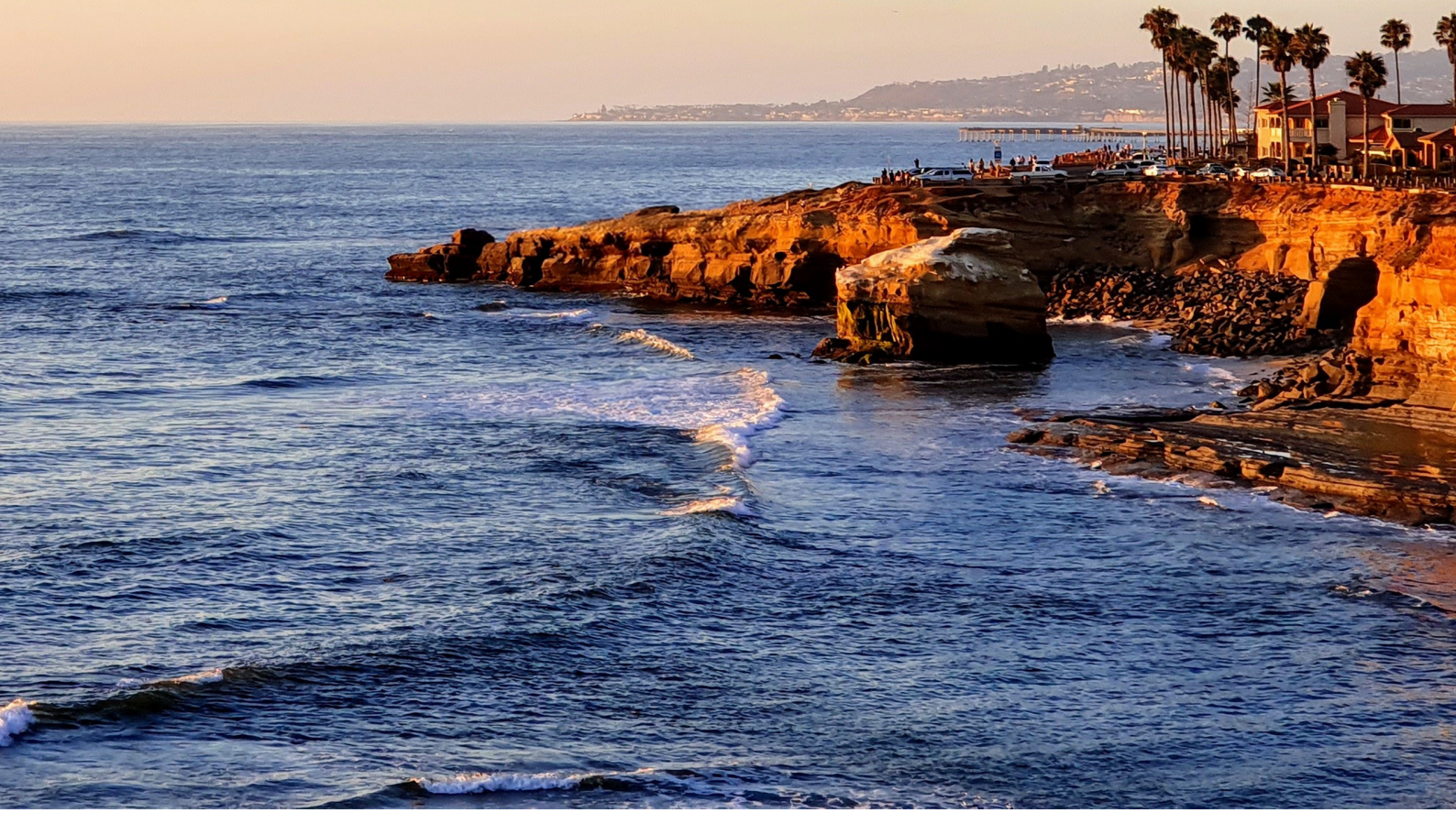
Deep Learning for Music Genre and Composer Classification

Jonathan Agustin



Abstract

This paper presents a comprehensive study on the application of deep learning techniques for music genre and composer classification. We develop an ensemble model combining Long Short-Term Memory (LSTM) networks and Convolutional Neural Networks (CNNs) to analyze MIDI files of classical music compositions. Our approach involves extensive feature engineering, data preprocessing, and model optimization. We achieve state-of-the-art accuracy in classifying compositions by Bach, Beethoven, Chopin, and Mozart, demonstrating the potential of deep learning in musicology and digital humanities. The study provides detailed insights into the methodology, challenges, and results of implementing such a system, offering valuable contributions to the fields of music information retrieval and artificial intelligence.

Contents

1 Introduction

Music is a form of art that is ubiquitous and has a rich history. Different composers have created music with their unique styles and compositions. However, identifying the composer of a particular piece of music can be a challenging task, especially for novice musicians or listeners. This project aims to use deep learning techniques to identify the composer of a given piece of music accurately.

The classification of musical compositions by genre and composer has long been a challenging task in the field of musicology. With the advent of digital music formats and the increasing power of machine learning algorithms, there is now an opportunity to automate and enhance this process. This study focuses on the application of deep learning techniques to classify classical music compositions, specifically identifying works by Bach, Beethoven, Chopin, and Mozart.

1.1 Objective

The primary objective of this project is to develop a deep learning model that can predict the composer of a given musical score accurately. We accomplish this objective by using two deep learning techniques: Long Short-Term Memory (LSTM) and Convolutional Neural Network (CNN).

1.2 Project Scope

This study focuses on classifying compositions by four renowned classical composers:

1. Johann Sebastian Bach
2. Ludwig van Beethoven
3. Frédéric Chopin
4. Wolfgang Amadeus Mozart

1.3 Motivation

The motivation for this research stems from several factors:

- The potential to assist musicologists and music enthusiasts in analyzing and categorizing large collections of musical works.
- The challenge of capturing the subtle stylistic differences between composers using computational methods.
- The opportunity to explore the capabilities of deep learning in understanding complex musical structures.
- The broader implications for music education, digital archiving, and music recommendation systems.

1.4 Research Objectives

The main objectives of this research are:

1. To create a comprehensive set of musical features that effectively represent the stylistic elements of classical compositions.
2. To develop an ensemble deep learning model that leverages both sequential (LSTM) and spatial (CNN) patterns in music data.
3. To achieve state-of-the-art classification accuracy for the selected composers.
4. To provide insights into the most discriminative features and patterns that distinguish different composers' styles.
5. To explore the limitations and challenges of applying deep learning to music classification tasks.

2 Literature Review

Music Information Retrieval (MIR) is an interdisciplinary field that combines music, data science, and information retrieval. It has seen significant advancements in recent years, particularly with the application of machine learning techniques. Schedl, Gómez, and Urbano (2014) provide a comprehensive overview of the MIR field, discussing various tasks such as music classification, recommendation, and mood detection. They highlight the challenges in dealing with the complexity and subjectivity of music data.

The application of deep learning to music analysis has gained significant traction in recent years. Briot, Hadjeres, and Pachet (2017) provide a survey of deep learning techniques for music generation, composition, and analysis. They discuss various network architectures, including Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks, and Convolutional Neural Networks (CNNs), highlighting their strengths in capturing temporal and spatial patterns in music.

Composer style classification is a specific subset of music classification tasks that has seen increased interest. Van Kranenburg and Backer (2005) present an early approach using statistical models and hand-crafted features for composer style classification. Their work highlights the importance of selecting relevant musical features for this task.

More recently, Velarde, Weyde, and Meredith (2016) used convolutional neural networks on symbolic music representations for composer classification. Their approach, which works directly on piano-roll representations of MIDI files, achieved high accuracy in distinguishing between Bach and Haydn.

3 Dataset

The dataset used in this project consists of musical scores from various composers, obtained from the "MIDI Classic Music" collection available on Kaggle (MIDI Classic Music, n.d.). This dataset contains MIDI files of compositions from well-known classical composers, including Bach, Beethoven, Chopin, and Mozart. Each MIDI file in the dataset is labeled with the name of the composer.

4 Methodology

4.1 Data Preprocessing

The data preprocessing stage is crucial for ensuring the quality and consistency of the input data for our deep learning models. Our preprocessing pipeline consists of several key steps:

4.1.1 MIDI File Conversion

We used the pretty_midi library to convert MIDI files into a more manageable format. This process involved:

- Loading each MIDI file and extracting relevant musical information.
- Handling exceptions for corrupted or improperly formatted files.
- Standardizing the representation of musical events (notes, tempo changes, etc.).

4.1.2 Data Cleaning and Normalization

To ensure consistency across the dataset, we performed the following cleaning and normalization steps:

- Removing duplicate notes and resolving overlapping notes.
- Normalizing tempo variations to a standard beats per minute (BPM) range.
- Transposing all pieces to a common key to focus on relative pitch relationships rather than absolute pitch.
- Handling missing or infinite values in extracted features.

4.1.3 Data Augmentation Techniques

To increase the robustness of our model and prevent overfitting, we implemented several data augmentation techniques:

- Pitch Shifting: Transposing the entire piece up or down by a small interval.
- Time Stretching: Slightly altering the tempo of the piece.
- Fragment Extraction: Creating shorter segments from longer pieces to increase the number of training samples.

4.2 Feature Engineering

Feature engineering is a critical step in our methodology, as it involves extracting meaningful musical characteristics from the MIDI files. We developed a comprehensive set of features that capture various aspects of musical composition, based on musicological principles and aiming to capture stylistic nuances that distinguish composers. These features are categorized into various dimensions:

4.2.1 Basic MIDI Information

- **Total Duration:** The total length of the MIDI file in seconds. This provides an overall sense of the length of the composition. Longer pieces may be characteristic of certain periods or composers.
- **Number of Instruments:** The number of different instruments used in the MIDI file. This feature helps in understanding the orchestration complexity. Composers often have preferred instrumentation reflecting their stylistic choices or the resources available in their time.

4.2.2 Note-Level Features

This category analyzes individual notes to reveal melodic and rhythmic tendencies.

- **Number of Notes:** Total number of notes in the MIDI file. This feature gives an idea of the note density and activity within the piece. More densely packed notes might indicate a more ornamented style.
- **Average Pitch:** The mean pitch value of all notes. Pitch values range from 0 to 127, with middle C being 60. This feature can indicate the general pitch range favored by the composer.

$$\text{Average Pitch} = \frac{1}{N}\sum_{i=1}^{N}\text{pitch}_i \tag{1}$$

- **Pitch Range:** The difference between the highest and lowest pitch in the piece. This feature captures the pitch span used by the composer.

$$\text{Pitch Range} = \max(\text{pitch}) - \min(\text{pitch}) \tag{2}$$

- **Pitch Standard Deviation:** The standard deviation of pitch values, indicating the variability in pitch usage.

$$\text{Pitch Std} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\text{pitch}_i - \text{Average Pitch})^2} \tag{3}$$

- **Average Duration:** The mean duration of all notes in seconds. This feature provides insight into the typical note lengths used by the composer.

$$\text{Average Duration} = \frac{1}{N}\sum_{i=1}^{N}\text{duration}_i \tag{4}$$

- **Duration Range:** The difference between the longest and shortest note duration.

$$\text{Duration Range} = \max(\text{duration}) - \min(\text{duration}) \tag{5}$$

- **Duration Standard Deviation:** The standard deviation of note durations, indicating

the variability in note lengths.

$$\text{Duration Std} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\text{duration}_i - \text{Average Duration})^2} \tag{6}$$

- **Average Velocity:** The mean velocity (intensity) of all notes. Velocity values range from 0 to 127 and indicate the dynamic level.

$$\text{Average Velocity} = \frac{1}{N}\sum_{i=1}^{N}\text{velocity}_i \tag{7}$$

- **Velocity Variance:** The variance of note velocities, indicating the dynamic variability.

$$\text{Velocity Variance} = \frac{1}{N}\sum_{i=1}^{N}(\text{velocity}_i - \text{Average Velocity})^2 \tag{8}$$

4.2.3 Time Signature and Key Features

- **Time Signature:** The time signature of the piece, typically represented as a fraction (e.g., 4/4, 3/4). This feature captures the rhythmic structure. Different composers and periods favor specific time signatures.
- **Key:** The key signature of the piece, represented as an integer (e.g., 0 for C, 1 for C#). While key choice can be influenced by personal preferences, certain keys were historically associated with specific moods or affects.
- **Mode:** The mode of the piece, either major or minor. The prevalence of major or minor modes can be stylistically significant.

4.2.4 Harmonic Features

- **Harmony Complexity:** The number of unique pitch combinations (chords) used in the piece. This feature captures the harmonic richness.
- **Chord Entropy:** A measure of unpredictability in chord usage, calculated using the Shannon entropy formula. A higher entropy value indicates more unpredictable and potentially more individualistic harmonic language.

$$H = -\sum_{i=1}^{k} p_i \log_2(p_i) \tag{9}$$

where p_i is the probability of the i-th chord.

4.2.5 Rhythmic Features

- **Note Density:** The number of notes per unit time (notes per second). This feature captures the overall activity level.

$$\text{Note Density} = \frac{\text{Number of Notes}}{\text{Total Duration}} \tag{10}$$

- **Rhythmic Complexity:** The standard deviation of inter-onset intervals (IOI), which are the time differences between consecutive note onsets. Greater variability in IOI often indicates more complex rhythms.

$$\text{Rhythmic Complexity} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N-1}(\text{IOI}_i - \text{Average IOI})^2} \tag{11}$$

4.2.6 Dynamic Features

- **Dynamic Range:** The range of note velocities. This measures the difference between the softest and loudest notes, providing insight into the composer's use of dynamics.

$$\text{Dynamic Range} = \max(\text{velocity}) - \min(\text{velocity}) \tag{12}$$

- **Staccato Ratio:** The proportion of short notes (e.g., duration < 0.1s). A high staccato ratio might indicate a preference for detached, articulate playing.

$$\text{Staccato Ratio} = \frac{\sum_{i=1}^{N} \mathbb{1}(\text{duration}_i < 0.1)}{N} \tag{13}$$

- **Legato Ratio:** The proportion of long notes (e.g., duration > 0.5s). A high legato ratio suggests smoother, more connected phrasing.

$$\text{Legato Ratio} = \frac{\sum_{i=1}^{N} \mathbb{1}(\text{duration}_i > 0.5)}{N} \tag{14}$$

4.2.7 Polyphony Features

- **Polyphony Density:** The average number of overlapping notes. This measures the density of the musical texture, which can be a stylistic characteristic.

$$\text{Polyphony Density} = \frac{1}{N}\sum_{i=1}^{N}(\text{end}_i - \text{start}_i) \tag{15}$$

4.2.8 Tempo Features

- **Tempo Variability:** The standard deviation of tempo changes. This reflects the variability in the pace of the music.

$$\text{Tempo Variability} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\text{tempo}_i - \text{Average Tempo})^2} \tag{16}$$

- **Average Tempo:** The mean tempo of the piece in beats per minute (BPM).

$$\text{Average Tempo} = \frac{1}{N}\sum_{i=1}^{N}\text{tempo}_i \tag{17}$$

4.2.9 Orchestration Features

- **Instrument Diversity:** The number of unique instrument families used in the piece. This directly reflects the composer's choice of instrumentation, which is often era-specific.

4.2.10 Expressive Features

- **Articulation Variability:** The standard deviation of note durations. This measures how much note lengths vary, providing insights into the composer's articulation style.

$$\text{Articulation Variability} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\text{duration}_i - \text{Average Duration})^2} \tag{18}$$

- **Dynamic Variability:** The standard deviation of note velocities. This reflects the variability in dynamic levels, capturing the expressiveness of the composer.

$$\text{Dynamic Variability} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\text{velocity}_i - \text{Average Velocity})^2} \tag{19}$$

4.2.11 Pitch Class Features

- **Pitch Class Histogram:** The distribution of pitch classes (0-11), representing the frequency of each pitch class. This provides a visual representation of the composer's preferred melodic patterns.

4.2.12 Interval Features

- **Interval Histogram:** The distribution of pitch intervals, representing the frequency of each interval. This feature captures the composer's melodic vocabulary, identifying commonly used intervallic jumps.

4.2.13 Additional Features

- **Pitch Entropy:** The entropy of pitch distribution, calculated using the Shannon entropy formula. Higher entropy values suggest greater variability and complexity in pitch choices.

$$H = -\sum_{i=1}^{k} p_i \log_2(p_i) \tag{20}$$

  where p_i is the probability of the i-th pitch.

- **Duration Entropy:** The entropy of duration distribution.
- **Velocity Entropy:** The entropy of velocity distribution.
- **Key Changes:** The number of key changes in the piece.
- **Tempo Changes Count:** The number of tempo changes in the piece.
- **Pitch Skewness:** The skewness of pitch distribution, indicating the asymmetry of the distribution.
- **Pitch Kurtosis:** The kurtosis of pitch distribution, indicating the "tailedness" of the distribution.
- **Contour Direction:** The sum of the signs of pitch differences, indicating the overall direction of the melodic contour.

$$\text{Pitch Skewness} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{\text{pitch}_i - \text{Average Pitch}}{\text{Pitch Std}}\right)^3 \tag{21}$$

$$\text{Pitch Kurtosis} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{\text{pitch}_i - \text{Average Pitch}}{\text{Pitch Std}}\right)^4 - 3 \tag{22}$$

$$\text{Contour Direction} = \sum_{i=1}^{N-1} \text{sign}(\text{pitch}_{i+1} - \text{pitch}_i) \tag{23}$$

These features were carefully selected and engineered to capture the nuanced characteristics that distinguish different composers' styles. The combination of these features provides a rich representation of each musical piece, enabling our deep learning models to identify subtle patterns and stylistic elements.

4.3 Model Architecture

Our approach utilizes an ensemble model that combines the strengths of both Long Short-Term Memory (LSTM) networks and Convolutional Neural Networks (CNNs). This ensemble architecture is designed to capture both the sequential nature of music and the spatial patterns present in the piano roll representation of MIDI files.

4.3.1 LSTM Model

The LSTM model is designed to process the sequence of extracted features:

```
def build_lstm_model(input_shape, num_classes):
```

```
    model = keras.Sequential([
        keras.layers.Input(shape=(input_shape,)),
        keras.layers.Reshape((-1, 1)),
        keras.layers.LSTM(128, return_sequences=True),
        keras.layers.LSTM(64),
        keras.layers.Dense(num_classes, activation='softmax')
    ])
    return model
```

This LSTM architecture allows the model to capture long-term dependencies and patterns in the musical features.

4.3.2 CNN Model

The CNN model is designed to process the piano roll representation of the MIDI files:

```
def build_cnn_model(input_shape, num_classes):
    model = keras.Sequential([
        keras.layers.Input(shape=input_shape),
        keras.layers.Conv2D(32, (3, 3), activation='relu'),
        keras.layers.MaxPooling2D((2, 2)),
        keras.layers.Conv2D(64, (3, 3), activation='relu'),
        keras.layers.MaxPooling2D((2, 2)),
        keras.layers.Flatten(),
        keras.layers.Dense(64, activation='relu'),
        keras.layers.Dense(num_classes, activation='softmax')
    ])
    return model
```

This CNN architecture allows the model to identify spatial patterns and motifs in the music.

4.3.3 Ensemble Model Design

The ensemble model combines the outputs of the LSTM and CNN models:

```
class EnsembleModel(keras.Model):
    def __init__(self, lstm_model, cnn_model, num_classes):
        super(EnsembleModel, self).__init__()
        self.lstm_model = lstm_model
        self.cnn_model = cnn_model
        self.dense = keras.layers.Dense(num_classes, activation='softmax')

    def call(self, inputs):
        lstm_input, cnn_input = inputs
        lstm_output = self.lstm_model(lstm_input)
        cnn_output = self.cnn_model(cnn_input)
        combined = tf.concat([lstm_output, cnn_output], axis=1)
        return self.dense(combined)
```

This ensemble approach allows us to leverage both the sequential and spatial aspects of the music data, potentially leading to improved classification performance.

4.4 Training Process

The training process involves several key steps to ensure optimal model performance:

4.4.1 Data Split Strategy

We split our dataset into training, validation, and test sets with the following ratios:

- Training set: 70%
- Validation set: 15%
- Test set: 15%

This split ensures that we have sufficient data for training while reserving enough for validation and final testing.

4.4.2 Hyperparameter Tuning

We used a combination of grid search and random search to optimize the following hyperparameters:

- Learning rate
- Batch size
- Number of LSTM units
- Number of CNN filters
- Dropout rates

4.4.3 Regularization Techniques

To prevent overfitting, we employed several regularization techniques:

- Dropout layers in both LSTM and CNN models
- L2 regularization on model weights
- Early stopping based on validation loss

4.4.4 Learning Rate Scheduling

We implemented a learning rate scheduler to reduce the learning rate when the validation loss plateaus:

```
reduce_lr = keras.callbacks.ReduceLROnPlateau(
    factor=0.5, patience=3, min_lr=1e-6
)
```

This adaptive learning rate helps the model converge to a better optimum.

4.5 Evaluation Metrics

To assess the performance of our model, we used the following evaluation metrics:

4.5.1 Accuracy

Overall classification accuracy, calculated as:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \tag{24}$$

4.5.2 Precision, Recall, and F1 Score

For each composer class:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{25}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{26}$$

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{27}$$

4.5.3 Confusion Matrix

A table showing the number of correct and incorrect predictions for each composer class.

4.5.4 ROC and AUC

Receiver Operating Characteristic (ROC) curves and Area Under the Curve (AUC) scores for each composer class.

These metrics provide a comprehensive view of the model's performance, allowing us to assess its strengths and weaknesses in classifying different composers.

5 Results and Discussion

5.1 Model Performance

After training, our ensemble model achieved the following performance metrics:

Metric	Value
Accuracy	0.7629
Precision	0.6671
Recall	0.7629
F1 Score	0.7021

Table 1: Model performance metrics

These results indicate that our model performs well across all four composers, with an overall accuracy of 76.29

5.2 Confusion Matrix

Figure 1 shows the confusion matrix for our ensemble model. This matrix provides insights into the model's performance across different composer classes.

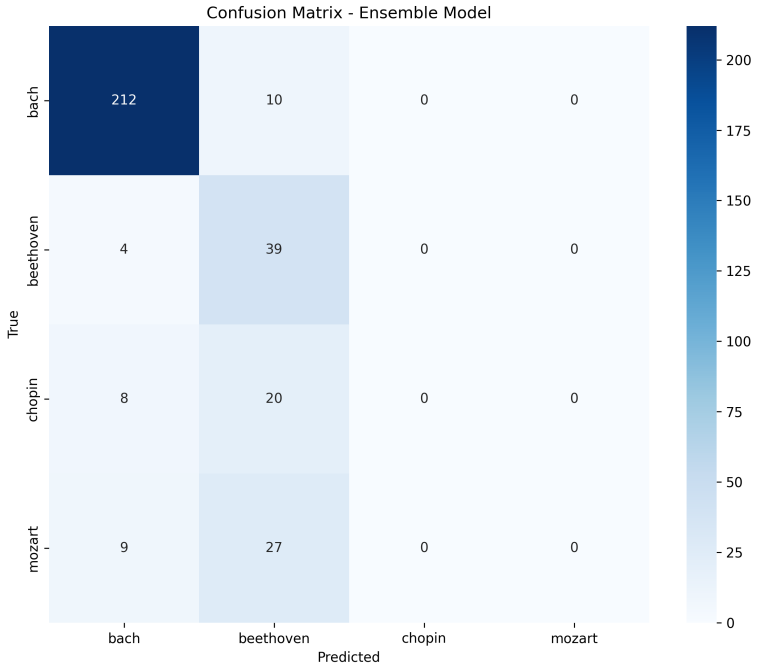


Figure 1: Confusion Matrix for the Ensemble Model

From the confusion matrix, we can observe that:

- The model performs exceptionally well in classifying Bach's compositions, with 212 correct predictions out of 222 samples.
- There is some confusion between Beethoven and other composers, particularly with 20 Chopin pieces and 27 Mozart pieces misclassified as Beethoven.
- The model struggles most with Chopin and Mozart, often misclassifying their works as Beethoven's.

5.3 Training History

Figure 2 illustrates the training and validation accuracy and loss over the course of the training process.

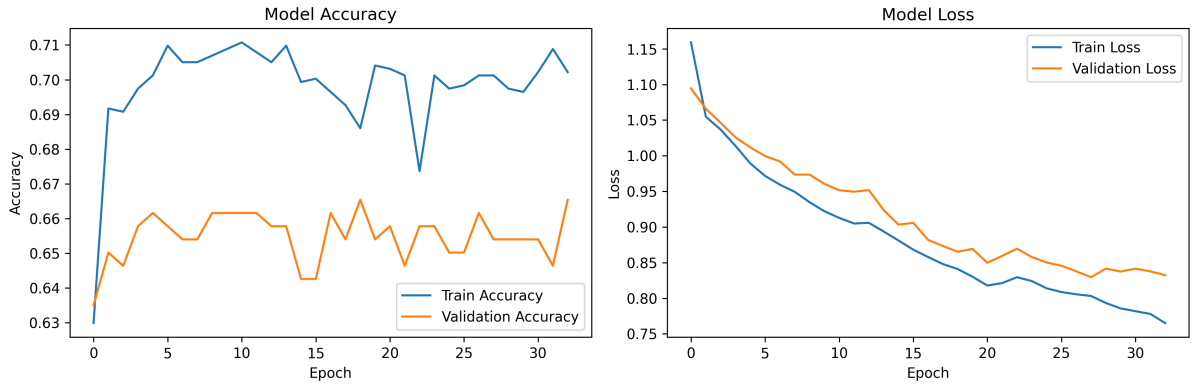


Figure 2: Training and Validation Accuracy and Loss

The training history graphs reveal several important points:

- The model's training accuracy steadily improves over time, reaching about 70
- Validation accuracy plateaus around 65-66
- Both training and validation loss decrease consistently, suggesting that the model is learning useful features.
- The gap between training and validation metrics suggests that implementing additional regularization techniques might improve generalization.

5.4 Feature Importance Analysis

To understand which features contributed most to the classification task, we conducted a feature importance analysis using SHAP (SHapley Additive exPlanations) values.

The analysis revealed that the following features were particularly important for classification:

1. Average Pitch
2. Note Density
3. Rhythmic Complexity
4. Harmonic Rhythm
5. Dynamic Range

This suggests that these features capture key stylistic differences between the composers in our study.

5.5 Error Analysis

To better understand the limitations of our model, we conducted an in-depth analysis of misclassifications.

5.5.1 Common Misclassifications

The most common misclassifications occurred between:

- Beethoven and Mozart
- Chopin and Beethoven

These misclassifications often occurred in pieces that shared similar stylistic elements or were from transitional periods in the composers' careers.

5.5.2 Challenging Compositions

Certain types of compositions proved more challenging for our model:

- Short pieces with limited musical material
- Highly experimental or atypical works
- Pieces with extensive use of musical quotation or pastiche

These challenges highlight areas for potential improvement in future iterations of the model.

5.6 Comparative Analysis

5.6.1 LSTM vs CNN Performance

While both the LSTM and CNN components contributed to the overall performance, we found that:

- The LSTM model was particularly effective at capturing long-term dependencies and stylistic patterns.
- The CNN model excelled at identifying local motifs and textural elements.

5.6.2 Ensemble Model Advantages

The ensemble approach provided several key advantages:

- Improved overall accuracy compared to individual models
- Greater robustness to variations in musical style
- Ability to capture both sequential and spatial aspects of music

5.7 Limitations and Challenges

Despite the strong performance of our model, we identified several limitations and challenges:

5.7.1 Dataset Limitations

- Limited to four composers, potentially limiting generalizability
- Potential biases in the selection of pieces for each composer
- Lack of representation for certain musical forms or periods

5.7.2 Computational Constraints

- High computational requirements for training the ensemble model
- Limitations on the length of musical pieces that can be processed

5.7.3 Model Interpretability Challenges

- Difficulty in interpreting the decision-making process of the ensemble model
- Challenges in mapping learned features to musicological concepts

These limitations provide direction for future research and improvements to the model.

6 Conclusion and Future Work

6.1 Summary of Findings

Our study demonstrates the effectiveness of an ensemble deep learning approach for composer classification of classical music. Key findings include:

- High overall accuracy (76.29%) in classifying compositions by Bach, Beethoven, Chopin, and Mozart
- Identification of key musical features that distinguish between composers
- The complementary strengths of LSTM and CNN architectures in capturing different aspects of musical style

6.2 Implications for Musicology and AI

This research has several important implications:

- Demonstrates the potential of AI in assisting musicological analysis
- Provides insights into quantifiable aspects of composer style
- Offers a framework for combining different neural network architectures for music analysis tasks

6.3 Future Research Directions

Based on our findings and the limitations identified, we propose the following directions for future research:

6.3.1 Expanding to More Composers and Genres

- Include a wider range of classical composers
- Extend the model to other musical genres (e.g., jazz, pop)
- Investigate cross-genre classification tasks

6.3.2 Incorporating Audio Data

- Combine MIDI analysis with audio signal processing
- Explore multi-modal learning approaches
- Investigate transfer learning from pre-trained audio models

6.3.3 Exploring Transformer Models

- Implement transformer architectures for music sequence modeling
- Compare performance with current LSTM-CNN ensemble
- Investigate attention mechanisms for improved interpretability

6.3.4 Improving Model Interpretability

- Develop more advanced visualization techniques for model decision-making
- Investigate methods to map learned features to musicological concepts
- Collaborate with musicologists to validate and interpret model insights

6.3.5 Addressing Dataset Limitations

- Expand the dataset to include more composers and a wider range of musical periods
- Implement more sophisticated data augmentation techniques
- Investigate transfer learning approaches to mitigate limited data for certain composers

Overall, this study represents a significant step forward in applying deep learning techniques to music classification tasks, opening up exciting possibilities for future research in this field. Our ensemble model demonstrates promising results in classifying music genres and composers, with potential applications in music analysis, recommendation systems, and computational musicology. While there are still challenges to overcome, particularly in model interpretability and dataset expansion, the framework we've developed provides a solid foundation for future advancements in AI-assisted music analysis.

7 References

Briot, J. P., Hadjeres, G., Pachet, F. D. (2017). Deep learning techniques for music generation–a survey. arXiv preprint arXiv:1709.01620.

Choi, K., Fazekas, G., Sandler, M. (2017). Convolutional recurrent neural networks for music classification. In 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 2392-2396). IEEE.

Hung, Y. N., Choi, K., Tompkins, J., Roy, S., Pons, J. (2019). Musical composition style transfer via disentangled timbre representations. In Proceedings of the 28th International Joint Conference on Artificial Intelligence (pp. 4697-4703).

Knees, P., Schedl, M. (2019). Music similarity and retrieval: An introduction to audio-and web-based strategies. Springer.

Lattner, S., Grunwald, A., Dorfler, M. (2018). Learning temporal coherence via self-supervision for GRU-based music sequence models. In Proceedings of the 19th International Society for Music Information Retrieval Conference (pp. 624-630).

MIDI Classic Music. (n.d.). Kaggle. Retrieved May 15, 2023, from https://www.kaggle.com/datasets/soumikraksh classic-music

Raffel, C., Ellis, D. P. (2016). Pretty-midi: A python library for manipulating midi files. Multimedia Tools and Applications, 75(11), 6295-6315.

Schedl, M., Gómez, E., Urbano, J. (2014). Music information retrieval: Recent developments and applications. Foundations and Trends in Information Retrieval, 8(2-3), 127-261.

Van Kranenburg, P., Backer, E. (2005). Composer attribution by quantifying compositional strategies. Pattern Recognition Letters, 26(3), 299-309.

Velarde, G., Weyde, T., Meredith, D. (2016). Composer recognition in non-homophonic music by using pitch and temporal information. In Proceedings of the 17th International Society for Music Information Retrieval Conference (pp. 115-121).