Arifa Kokab

Student ID: 009588754

University of San Diego

Shiley-Marcos School of Engineering

AAI-511

Professor Kahila Mokhtari Jadid

Project Report: Hybrid LSTM-CNN Model for Composer Classification

Team Final Project

TABLE OF CONTENTS

Abstract

This project aimed to develop a hybrid deep learning model combining Long Short-Term Memory (LSTM) and Convolutional Neural Networks (CNN) to classify musical compositions according to their composers. The dataset consisted of MIDI files from four classical composers: Johann Sebastian Bach, George Frideric Handel, Frédéric Chopin, and Wolfgang Amadeus Mozart. The model was designed to capture both temporal dependencies and local patterns in the musical sequences. The hybrid model achieved an accuracy of 54.55%, a precision of 70.83%, and a recall of 54.55%. This report documents the methodology, data preprocessing steps, feature extraction techniques, model architecture, training process, and evaluation metrics, with a focus on reproducibility and future improvements.

Introduction

Composer classification presents a significant challenge in the field of music analysis, as it involves distinguishing between the unique styles and techniques employed by different composers. Each composer is known for a distinct musical signature, which can be identified through various elements such as pitch sequences, recurring motifs, harmonic structures, and rhythmic patterns. These elements combine to create a composer's unique voice, making the task of classification complex and requiring sophisticated analytical methods.

Traditionally, musicologists and analysts have relied on manual examination to identify these stylistic traits, but with the advent of deep learning, it is now possible to automate this process and potentially achieve even more nuanced insights. Deep learning models are particularly well-suited for this task due to their ability to learn from vast amounts of data and extract meaningful patterns without explicit feature engineering.

This project investigates the application of a hybrid deep learning model that integrates two powerful architectures: Long Short-Term Memory (LSTM) networks and Convolutional Neural Networks (CNNs). Each of these architectures brings distinct strengths to the task of composer classification.

The LSTM component of the model is designed to handle the sequential nature of musical data. Music is inherently temporal, with each note or chord often depending on what came before it. LSTM networks are capable of capturing these dependencies over time, making them particularly effective for analyzing the flow of musical compositions. By processing sequences of notes, the LSTM layers in the model can learn to recognize the progression of a melody or harmony that is characteristic of a particular composer.

On the other hand, the CNN component of the model is adept at identifying local patterns within the pitch sequences. CNNs are typically used in image processing, where they excel at detecting features such as edges, textures, and shapes. When applied to music, CNNs can similarly detect motifs, chord progressions, or other small-scale patterns that recur throughout a piece of music. These local features are often the building blocks of a composer's style, and their identification is crucial for accurate classification.

By combining LSTM and CNN architectures into a single hybrid model, this project aims to leverage the strengths of both approaches. The LSTM layers capture the global, sequential aspects of the music, while the CNN layers focus on local patterns that might be indicative of a composer’s stylistic signature. This hybrid approach allows for a more comprehensive analysis of the musical data, potentially leading to more accurate and robust composer classification.

In summary, this project seeks to push the boundaries of composer classification by employing a hybrid LSTM-CNN model. By integrating these two architectures, the model is equipped to analyze both the temporal and local aspects of musical compositions, providing a deeper understanding of the unique ### Introduction

Composer classification presents a significant challenge in the field of music analysis, as it involves distinguishing between the unique styles and techniques employed by different composers. Each composer is known for a distinct musical signature, which can be identified through various elements such as pitch sequences, recurring motifs, harmonic structures, and rhythmic patterns. These elements combine to create a composer's unique voice, making the task of classification complex and requiring sophisticated analytical methods.

Traditionally, musicologists and analysts have relied on manual examination to identify these stylistic traits, but with the advent of deep learning, it is now possible to automate this process and potentially achieve even more nuanced insights. Deep learning models are particularly well-suited for this task due to their ability to learn from vast amounts of data and extract meaningful patterns without explicit feature engineering.

This project investigates the application of a hybrid deep learning model that integrates two powerful architectures: Long Short-Term Memory (LSTM) networks and Convolutional Neural Networks (CNNs). Each of these architectures brings distinct strengths to the task of composer classification.

The LSTM component of the model is designed to handle the sequential nature of musical data. Music is inherently temporal, with each note or chord often depending on what came before it. LSTM networks are capable of capturing these dependencies over time, making them particularly effective for analyzing the flow of musical compositions. By processing sequences of notes, the LSTM layers in the model can learn to recognize the progression of a melody or harmony that is characteristic of a particular composer.

On the other hand, the CNN component of the model is adept at identifying local patterns within the pitch sequences. CNNs are typically used in image processing, where they excel at detecting features such as edges, textures, and shapes. When applied to music, CNNs can similarly detect motifs, chord progressions, or other small-scale patterns that recur throughout a piece of music. These local features are often the building blocks of a composer's style, and their identification is crucial for accurate classification.

By combining LSTM and CNN architectures into a single hybrid model, this project aims to leverage the strengths of both approaches. The LSTM layers capture the global, sequential aspects of the music, while the CNN layers focus on local patterns that might be indicative of a composer's stylistic signature. This hybrid approach allows for a more comprehensive analysis of the musical data, potentially leading to more accurate and robust composer classification.

In summary, this project seeks to push the boundaries of composer classification by employing a hybrid LSTM-CNN model. By integrating these two architectures, the model is equipped to analyze both the temporal and local aspects of musical compositions, providing a deeper understanding of the unique elements that define each composer's style. The outcome is expected to offer valuable insights into how different musical elements contribute to the distinctiveness of a composer's work, with potential applications in musicology, automated music analysis, and educational tools.elements that define each composer's style. The outcome is expected to offer valuable insights into how different musical elements contribute to the distinctiveness of a composer's work, with potential applications in musicology, automated music analysis, and educational tools.

Methodology

The methodology of this project was meticulously designed to ensure a systematic and comprehensive approach to developing and evaluating the hybrid LSTM-CNN model for composer classification. Each stage of the process was carefully planned and executed to maximize the model's effectiveness and to ensure that the results were both reliable and reproducible. The key stages of this methodology included data collection, preprocessing, feature extraction, model architecture design, model training, and evaluation.

1. Data Collection

The initial step in the methodology was data collection, a critical phase that serves as the foundation for any machine learning project. The quality, quantity, and relevance of the data directly impact the effectiveness of the models, making data collection a pivotal aspect of the project. In this case, the data consisted of MIDI (Musical Instrument Digital Interface) files, which are digital representations of musical compositions. These MIDI files were selected to represent works from four of the most renowned classical composers: Johann Sebastian Bach, George Frideric Handel, Frédéric Chopin, and Wolfgang Amadeus Mozart.

The selection of these particular composers was a deliberate choice, driven by the distinctive and identifiable styles that each composer is known for. Bach's intricate counterpoint, Handel's grand and expressive themes, Chopin's lyrical and technically demanding piano works, and Mozart's elegant and balanced compositions each present unique characteristics that pose both a challenge and an opportunity for classification using deep learning models. This diversity of styles provided a robust testing ground for the models, ensuring that they could learn to differentiate between complex and nuanced musical patterns.

Organization of Data

Once the composers were selected, the MIDI files were organized into a structured directory within a Google Drive folder, with separate subfolders for each composer. For example, all MIDI files corresponding to Bach's compositions were placed in a folder labeled "Bach," while Handel's compositions were placed in a folder labeled "Handel," and so forth. This systematic organization was essential for the subsequent steps in the project, as it enabled efficient and error-free processing of the data.

The structure of the dataset allowed for streamlined preprocessing and feature extraction, as each subfolder could be processed independently, ensuring that the correct labels (i.e., composer names) were associated with each piece of music. This organization also facilitated the implementation of any necessary data transformations, such as converting MIDI files into arrays of note pitches, by providing a clear and consistent framework within which these operations could be performed.

Sourcing and Reproducibility

The MIDI files were sourced from publicly available repositories, which are online platforms that provide access to a vast array of musical data. These repositories are invaluable resources for researchers and developers working in the field of music technology, as they offer a wide range of compositions across various genres and time periods. By sourcing the data from such repositories, the project ensured that it adhered to principles of transparency and reproducibility.

Reproducibility is a cornerstone of scientific research and machine learning development. By using publicly available datasets, this project allows others to replicate the study, validate the findings, and potentially build upon the work. Other researchers or developers can access the same MIDI files, apply the same preprocessing techniques, and compare the performance of different models using identical data. This openness not only strengthens the validity of the results but also fosters collaboration and innovation within the research community.

Furthermore, the use of publicly available data reduces barriers to entry for those who wish to explore similar projects. Whether they are academic researchers, hobbyists, or industry professionals, individuals can access the necessary data without the need for proprietary resources or specialized collections, making the project more inclusive and accessible.

Summary

In summary, the data collection phase of this project was meticulously planned and executed to ensure that the foundational data was both high-quality and well-organized. By selecting compositions from four distinct and well-known classical composers, the project was set up to tackle a challenging yet feasible classification task. The organization of the MIDI files into composer-specific folders facilitated smooth processing in later stages of the project, and the use of publicly available data sources ensured that the work was transparent, reproducible, and accessible to the broader research community. This careful approach to data collection laid the groundwork for the successful development and evaluation of the deep learning models that followed.

2. Data Preprocessing

After collecting the data, the next critical step in the project was to preprocess it into a format that could be effectively used as input for the hybrid LSTM-CNN model. The preprocessing phase is essential in any machine learning project, as it transforms raw data into a structured format that the model can interpret and learn from. For this project, the preprocessing was particularly important due to the unique nature of the MIDI files and the specific requirements of the hybrid model.

MIDI (Musical Instrument Digital Interface) files are a standard digital representation of music that encodes various aspects of a musical performance, including note pitch, duration, velocity, and instrument data. These files offer a rich source of information, capturing the nuances of how each note is played, the dynamics of the performance, and the choice of instruments. However, given the focus of this project on composer classification, it was necessary to identify the most relevant aspects of this data that would contribute to distinguishing between different composers.

For the purpose of this project, the focus was placed specifically on the pitch sequences within the MIDI files. Pitch sequences represent the sequence of notes in terms of their frequencies, which are fundamental to the melodic and harmonic structure of music. Melodies, chord progressions, and other musical elements that define a composer's style are inherently tied to the pitch sequences. Therefore, by focusing on pitch, the model could learn the patterns and relationships between notes that are characteristic of each composer.

To extract these pitch sequences, the `pretty_midi` library was employed. `pretty_midi` is a powerful tool for working with MIDI data in Python, allowing for the efficient parsing and manipulation of musical information. The preprocessing process began with parsing each MIDI file to access its note data. This involved iterating through the notes played by each instrument in the composition and recording their pitch values. The pitch values were then assembled into sequences, where each sequence represented the ordered pitches of a composition as they occurred in time.

However, the compositions varied significantly in length due to differences in their complexity and duration. Some compositions were brief, containing only a few notes or phrases, while others were extensive, with complex melodic developments. This variability in sequence length posed a challenge for the LSTM component of the model, which requires input sequences to be of consistent length. To address this, a padding strategy was implemented.

Padding is a common technique used in sequence modeling where sequences are extended to a uniform length by adding extra values—typically zeros—at the end of shorter sequences. In this project, padding involved appending zeros to the end of pitch sequences that were shorter than the longest sequence in the dataset. This ensured that all sequences were of equal length, allowing the model to process them uniformly without bias toward longer compositions. Importantly, the zeros used for padding did not represent actual pitches, so they did not interfere with the model's ability to learn from the meaningful parts of the sequence.

Once the sequences were padded, they were reshaped as needed to fit the specific input requirements of the hybrid LSTM-CNN model. The LSTM layers expected 3D input in the form of `(batch_size, sequence_length, features)`, where `sequence_length` was the length of the padded pitch sequence and `features` was set to 1, representing the single dimension of pitch. After passing through the LSTM layers, the output was further reshaped to accommodate the input format required by the CNN layers, which processed the data in 2D or 3D format to detect local patterns within the sequences.

This preprocessing step was crucial for transforming the raw MIDI data into a format that could be effectively used by the hybrid model. By focusing on pitch sequences, the preprocessing ensured that the model received the most relevant information for composer classification. The use of padding allowed the model to handle compositions of varying lengths, and the careful reshaping of the data ensured compatibility with the complex architecture of the hybrid LSTM-CNN model. This thorough and systematic approach to preprocessing laid the groundwork for the model to learn and generalize from the data, ultimately contributing to the effectiveness of the composer classification task.

3. Feature Extraction

In this project, feature extraction was a critical process aimed at leveraging the advanced capabilities of the Long Short-Term Memory (LSTM) and Convolutional Neural Network (CNN) architectures to automatically learn and extract the most relevant features from the pitch sequences of musical compositions. Unlike traditional approaches that often require manual feature engineering, where experts explicitly define the characteristics to be extracted, this project utilized the power of deep learning to let the model autonomously discover patterns in the data that were most indicative of each composer's unique style. This approach not only streamlines the process but also has the potential to uncover subtle, complex features that may be difficult to identify manually.

LSTM Layers: Capturing Temporal Dependencies

The LSTM layers played a pivotal role in the feature extraction process by focusing on the temporal dynamics of the music. Music, by its very nature, is a sequential art form where each note, chord, or rhythmic element is influenced by what precedes and often by what follows it. This temporal dependency is especially important in understanding a composer's style, as the progression of notes and the evolution of musical ideas over time are key characteristics that define their work.

LSTM networks are specifically designed to handle such sequential data. They do so by maintaining a memory of previous inputs, which allows them to capture long-range dependencies and recognize patterns that unfold over time. In the context of this project, the LSTM layers analyzed the pitch sequences, learning how melodies and harmonies developed throughout a composition. For instance, the LSTM layers could identify how a particular composer might use a specific sequence of intervals or how they structure a melody to lead to a harmonic resolution. By capturing these temporal patterns, the LSTM layers were able to discern the flow and structure of the music, which are crucial for accurately identifying a composer's signature style.

This capability of LSTMs to retain information over long sequences is particularly valuable in music analysis, where understanding the relationship between distant notes can be as important as recognizing immediate, local patterns. The LSTM layers effectively acted as the model's "memory," allowing it to keep track of the development of musical ideas across the entire composition, and to use this information to make informed predictions about the composer.

CNN Layers: Detecting Local Patterns

While the LSTM layers focused on the global, temporal aspects of the music, the CNN layers were tasked with identifying local patterns within the pitch sequences. Convolutional Neural Networks are well-known for their ability to detect features in spatial data, such as images, by applying convolutional filters that move across the data to identify specific patterns. When adapted for sequential data like music, CNNs can be similarly effective at spotting localized features, such as recurring motifs, chord progressions, or rhythmic patterns that are characteristic of a particular composer.

In this project, the CNN layers applied convolutional filters across the pitch sequences to detect these local features. Each filter was designed to pick up on specific patterns within the sequence, such as a particular combination of pitches that might constitute a motif, or a sequence of chords that are frequently used by a composer. By detecting these patterns, the CNN layers were able to capture the finer details of the music that contribute to a composer's distinct style.

For example, a CNN filter might learn to recognize a characteristic chord progression that Chopin often uses in his nocturnes, or a particular melodic contour that is prevalent in Bach's fugues. These local features are often the building blocks of a composer's style, and by identifying them, the CNN layers provided the model with a detailed understanding of the musical composition.

The CNN's ability to focus on specific sections of the pitch sequence allowed it to complement the LSTM's broader temporal analysis. While the LSTM layers were concerned with how the music evolved over time, the CNN layers honed in on the small-scale patterns that are repeated within a piece or across different compositions by the same composer. This dual approach enabled the model to construct a comprehensive representation of the music, taking into account both the macro-level structure and the micro-level details.

Synergy of LSTM and CNN: Building a Robust Understanding

The combination of LSTM and CNN layers in this hybrid model created a powerful feature extraction process that leveraged the strengths of both architectures. The LSTM layers provided the model with a deep understanding of the temporal dependencies in the music, allowing it to capture the flow and development of musical ideas. Meanwhile, the CNN layers added another layer of analysis by detecting local patterns that contribute to a composer's style.

This synergy between global (LSTM) and local (CNN) feature extraction was critical in building a robust understanding of the musical data. By integrating these two approaches, the model was able to learn a rich set of features that spanned both the temporal and spatial dimensions of the music. This comprehensive feature set enabled the model to make accurate predictions about the composer of a given piece, as it could draw on a detailed and nuanced understanding of the music's structure and style.

In summary, the feature extraction process in this project was highly effective in utilizing the capabilities of LSTM and CNN architectures to automatically learn the most relevant patterns in the pitch sequences. By capturing both the temporal dynamics and local patterns in the music, the model was able to develop a deep and comprehensive understanding of each composer's style, leading to more accurate and reliable classification results. This approach not only simplified the feature extraction process but also enhanced the model's ability to generalize to new compositions, making it a powerful tool for composer classification.

4. Model Architecture

The architecture of the hybrid LSTM-CNN model was meticulously designed to capitalize on the strengths of both Long Short-Term Memory (LSTM) networks and Convolutional Neural Networks (CNNs), integrating these two powerful deep learning architectures to create a model capable of accurately classifying musical compositions by composer. This hybrid approach was chosen to leverage the unique capabilities of LSTMs in handling sequential data and CNNs in detecting localized patterns, thereby providing a comprehensive analysis of the musical data.

Input and Embedding Layers

The model architecture began with an input layer that accepted the preprocessed and padded sequences of note pitches. These sequences represented the core musical information extracted from the MIDI files, with each sequence corresponding to the ordered pitches in a composition. Given that pitch values are discrete, representing them as raw numerical values would limit the model's ability to understand the relationships between different notes. To address this, the input sequences were passed through an embedding layer.

The embedding layer played a crucial role by converting the discrete pitch values into dense vector representations. This transformation is analogous to how words are embedded in natural language processing tasks, where words are mapped to dense vectors in a continuous vector space. In the context of music, this embedding allowed the model to capture nuanced relationships between pitches—such as the similarity between pitches that are close in frequency or that function similarly within a scale or chord. By embedding the pitch sequences, the model could learn a richer, more expressive representation of the musical data, enabling it to better understand the underlying structure of the compositions.

LSTM Layers: Capturing Temporal Dependencies

At the core of the model were two LSTM layers, which were responsible for processing the pitch sequences in a manner that preserved their temporal order. The LSTM layers were specifically chosen for their ability to handle sequential data, where the order of elements is crucial. Music is inherently temporal, with each note or chord depending on its predecessors to create a coherent melodic or harmonic progression. The LSTM layers in the model were designed to capture these temporal dependencies, effectively learning how the music unfolds over time.

The first LSTM layer processed the input sequence, passing its output to the second LSTM layer. The use of two stacked LSTM layers allowed the model to build a deeper understanding of the sequence, capturing both short-term and long-term dependencies. For example, the LSTM layers could learn how a particular motif introduced early in a composition might be revisited or developed later, a common technique in classical music. This capability was essential for identifying the unique compositional techniques employed by each composer, which often manifest in how musical ideas are developed over time.

CNN Layers: Detecting Local Patterns

Following the LSTM layers, the output was reshaped and passed through a series of convolutional layers. The CNN layers were tasked with detecting significant local patterns within the pitch sequences. While the LSTM layers focused on the temporal aspects of the music, the CNN layers were designed to identify spatial features—localized patterns such as motifs, chord progressions, or specific pitch intervals that might recur within a piece or across different compositions by the same composer.

Each convolutional layer applied a set of filters to the reshaped output from the LSTM layers. These filters acted as pattern detectors, scanning across the sequence to identify features that were characteristic of a particular composer’s style. For example, a CNN filter might detect a recurring sequence of notes that forms a motif frequently used by Mozart, or a specific chord progression that is typical of Chopin’s compositions. By applying multiple filters, the CNN layers could extract a rich set of features from the music, each contributing to the model’s understanding of the composition.

The combination of LSTM and CNN layers provided the model with a powerful dual approach to feature extraction. While the LSTM layers captured the global structure of the music, the CNN layers focused on the local details, allowing the model to build a comprehensive representation of each composition.

Dense Layers and Output Layer

After the CNN layers had extracted the relevant features, the output was flattened and passed through fully connected (dense) layers. These dense layers synthesized the features extracted by the LSTM and CNN layers, combining the temporal and spatial information to make the final classification decision. The dense layers acted as the final decision-making part of the model, where the learned features were used to predict the composer of the musical piece.

The output layer used a softmax activation function to produce a probability distribution over the four composer classes: Bach, Handel, Chopin, and Mozart. This probability distribution indicated the model's confidence in each class, allowing it to predict the most likely composer for each musical composition. The softmax function ensured that the sum of the probabilities across all classes equaled one, making it straightforward to interpret the model's predictions.

Summary

The carefully designed architecture of the hybrid LSTM-CNN model, combined with a rigorous training process, enabled the model to effectively classify musical compositions by composer. The integration of LSTM and CNN layers allowed the model to capture both temporal and local patterns in the music, leading to a deeper and more nuanced understanding of each composer's style. Through the use of the Adam optimizer, sparse categorical cross-entropy loss, and early stopping, the model was trained to achieve a balance between learning and generalization, resulting in a robust and accurate classifier for musical compositions.

5. Model Training and Optimization

Training the hybrid LSTM-CNN model was a critical step in ensuring its ability to accurately classify musical compositions. The training process involved optimizing the model's parameters so that it could learn to recognize the patterns and features that distinguish one composer's style from another.

Optimizer and Loss Function

The model was trained using the Adam optimizer, a popular choice in deep learning due to its efficiency and adaptive learning rate. Adam combines the advantages of two other widely used optimizers, AdaGrad and RMSProp, making it well-suited for handling sparse gradients and non-stationary objectives. This optimizer was particularly effective for this project, where the model needed to learn from complex and varied musical data.

The loss function used during training was sparse categorical cross-entropy. This loss function is appropriate for multi-class classification tasks, where the goal is to correctly assign each input to one of several possible classes—in this case, the four composers. Sparse categorical cross-entropy measures the difference between the predicted probability distribution and the actual class label, encouraging the model to improve its predictions over time.

Training and Validation

To ensure that the model could generalize well to new, unseen compositions, the dataset was divided into training and validation sets. The training set was used to fit the model, with the model learning from the examples provided. The validation set served as a benchmark to monitor the model's performance during training. By evaluating the model on the validation set after each epoch, the project could assess how well the model was learning and make adjustments if necessary.

The model was trained over multiple epochs, with each epoch consisting of a full pass through the training data. Training over multiple epochs allowed the model to gradually refine its understanding of the data, improving its ability to classify compositions accurately.

Early Stopping to Prevent Overfitting

One of the key challenges in training deep learning models is preventing overfitting, where the model becomes too specialized to the training data and fails to generalize to new examples. To address this, early stopping was implemented during training. Early stopping is a technique that halts the training process if the model's performance on the validation set stops improving, indicating that further training would only lead to overfitting.

By monitoring the validation loss and stopping the training process when no further improvement was observed, the project ensured that the model did not overfit to the training data. This approach helped the model maintain its ability to generalize to unseen compositions, making it more reliable and effective in real-world applications.

<u>6. Model Evaluation</u>

Upon completing the training process, the performance of the hybrid LSTM-CNN model was rigorously evaluated using three critical metrics: accuracy, precision, and recall. These metrics were carefully selected to provide a comprehensive and multidimensional assessment of the model's effectiveness in classifying musical compositions by composer. Each metric offers a different perspective on the model's performance, allowing for a thorough analysis of its strengths and potential areas for improvement.

Accuracy: Overall Correctness of Predictions

Accuracy is one of the most straightforward and commonly used metrics in classification tasks. It measures the proportion of correct predictions made by the model out of the total number of predictions. In the context of this project, accuracy was calculated as the ratio of correctly identified composers to the total number of musical compositions in the test set.

An accuracy score provides a general overview of the model's performance, indicating how often the model is correct when classifying a piece of music. However, while accuracy is an important indicator of overall correctness, it does not account for the nuances of the classification task, such as how the model performs when dealing with imbalanced classes or when making false positive and false negative predictions. For this reason, accuracy was complemented by precision and recall to gain deeper insights into the model's behavior.

Precision: Reliability of Positive Predictions

Precision delves deeper into the model's ability to make reliable positive predictions. Specifically, precision measures the proportion of true positive predictions (correctly identified composers) among all the positive predictions made by the model (including both true positives and false positives). In simpler terms, precision answers the question: "When the model predicts a specific composer, how often is it correct?"

This metric is particularly valuable in scenarios where false positives—incorrectly labeling a composition as belonging to a certain composer—are of concern. For instance, if the model frequently predicts "Bach" but often does so incorrectly, it would have low precision for the "Bach" class. High precision indicates that the model is conservative in its predictions, preferring to make fewer but more accurate positive identifications. This is crucial in ensuring that when the model does make a prediction, it is likely to be correct, thus enhancing the model's reliability and trustworthiness.

Recall: Sensitivity to True Positives

Recall, also known as sensitivity or true positive rate, complements precision by evaluating the model's ability to identify all true positive instances among the actual positives. It answers the question: "Out of all the compositions that truly belong to a specific composer, how many did the model correctly identify?"

High recall indicates that the model is effective at capturing the correct class labels and that it is less likely to miss true positive instances. This metric is particularly important in contexts where it is critical not to overlook any true instances of a class. In the context of composer classification, high recall ensures that the model successfully identifies as many compositions by each composer as possible, even if it occasionally makes some incorrect predictions.

Comprehensive Understanding of Model Performance

Together, accuracy, precision, and recall provide a multifaceted evaluation of the model’s performance. Accuracy offers a broad measure of overall correctness, while precision and recall provide deeper insights into how the model balances the trade-offs between making accurate positive predictions and capturing all true instances of each class.

By analyzing these metrics, the project was able to identify the model's strengths, such as its ability to make reliable predictions (as indicated by high precision), and potential areas for improvement, such as enhancing its ability to capture all true positives (as indicated by recall). This comprehensive understanding of the model's performance was critical in determining the effectiveness of the hybrid LSTM-CNN architecture for composer classification and in guiding future efforts to refine and improve the model.

In particular, these metrics allowed the project team to assess whether the model was more prone to false positives (which would lower precision) or false negatives (which would lower recall), and to adjust the model's architecture, training process, or data preprocessing techniques accordingly. This iterative process of evaluation and refinement is essential in developing a robust, high-performing model that can accurately and reliably classify musical compositions by composer across a wide range of examples.

7. Results

The hybrid LSTM-CNN model, designed to classify musical compositions by composer, yielded the following performance metrics:

- **Accuracy**: 0.5454545454545454
- **Precision**: 0.7083333333333334
- **Recall**: 0.5454545454545454

These results provide valuable insights into the model's strengths and limitations.

Accuracy: Moderate Overall Correctness

The accuracy score of approximately 54.55% indicates that the model correctly identified the composer for just over half of the compositions in the test set. While this demonstrates that the model was able to learn and apply some of the distinguishing features of the composers, it also suggests that there is significant room for improvement. An accuracy of 54.55% indicates that the model might be struggling with correctly classifying compositions that are stylistically similar or that it may not have fully captured the nuances that differentiate each composer's style.

Precision: High Reliability in Positive Predictions

The model's precision score of approximately 70.83% is notably higher than its accuracy, indicating that when the model predicts a specific composer, it is correct around 71% of the time. This suggests that the model is relatively conservative in its predictions, meaning it is more likely to make a correct positive identification when it does choose to label a composition as belonging to a certain composer. This high precision is an encouraging sign, as it implies that the model has learned to identify key features that are strongly indicative of each composer's style, reducing the likelihood of false positives.

Recall: Moderate Sensitivity to True Positives

The recall score matches the accuracy at approximately 54.55%, indicating that the model correctly identified just over half of the true positive instances among the actual positives. This score suggests that while the model is effective at making correct predictions (as evidenced by the precision), it may be missing a significant portion of the true instances of each composer. This could imply that the model is underfitting to some degree or that it is not fully capturing the breadth of each composer's style, leading it to overlook some compositions that it should have identified correctly.

<u>Conclusion</u>

Key Findings

The performance metrics reveal a hybrid model that has demonstrated its ability to differentiate between composers to a certain extent, particularly when it comes to making reliable positive predictions. However, the moderate accuracy and recall scores suggest that the model has difficulty consistently recognizing all compositions by each composer. This indicates that while the model has learned to recognize certain distinctive features of each composer's style, it may not have fully generalized this understanding across the entire dataset.

Conclusions and Future Directions

The results of the hybrid LSTM-CNN model show promise in the area of composer classification, particularly in its ability to make reliable predictions (as indicated by the precision score). However, the moderate accuracy and recall suggest that the model's current architecture and training process have limitations that need to be addressed to improve its overall performance.

Potential Improvements and Extensions

1. **Enhanced Feature Engineering**:
    - While the current model focuses primarily on pitch sequences, future iterations could incorporate additional musical features such as rhythm, harmony, dynamics, and articulation. These elements could provide the model with a richer understanding of each composer's style, potentially improving accuracy and recall.
2. **Alternative Deep Learning Architectures**:
    - Exploring other deep learning architectures, such as Transformer models, could help improve the model's ability to capture long-range dependencies and subtle stylistic nuances that the current LSTM-CNN hybrid may be missing.
3. **Data Augmentation**:
    - Implementing data augmentation techniques, such as transposing compositions to different keys or varying the tempo, could increase the diversity of the training data.

This could help the model generalize better to new compositions and improve its recall by exposing it to a broader range of musical variations.

4. **Increased Dataset Size**:
    - Expanding the dataset to include more composers and a greater number of compositions per composer could provide the model with more examples to learn from, potentially improving its ability to generalize and recognize true positives.
5. **Hyperparameter Tuning**:
    - Further tuning of the model's hyperparameters, such as the number of layers, learning rate, and batch size, could help optimize the model's performance and address any underfitting or overfitting issues.

Final Summary

This project successfully developed a hybrid LSTM-CNN model for the classification of musical compositions by composer. The model demonstrated moderate accuracy and recall, with higher precision, indicating that it has learned to make reliable predictions but still struggles with fully capturing the diversity of each composer's style. The findings highlight the potential of deep learning in music classification while also pointing to areas where further research and development could lead to significant improvements. Future work could focus on enhancing feature extraction, experimenting with alternative architectures, and expanding the dataset to build on the foundation established by this project. With these improvements, the model could achieve higher accuracy and recall, making it a more robust and effective tool for music analysis and composer classification.

References

Chollet, F. (2017). *Deep learning with Python*. Manning Publications.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. https://arxiv.org/abs/1412.6980

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825-2830. http://jmlr.org/papers/v12/pedregosa11a.html

Raffel, C., Ellis, D. P. W. (2016). *MIDI mining: Extracting and analyzing symbolic music from the web.* In *16th International Society for Music Information Retrieval Conference (ISMIR)*, Málaga, Spain, October 26-30, 2015. https://craffel.github.io/pretty-midi/

TensorFlow. (n.d.). *TensorFlow: An end-to-end open-source machine learning platform*. https://www.tensorflow.org/

Van Der Walt, S., Colbert, S. C., & Varoquaux, G. (2011). The NumPy array: A structure for efficient numerical computation. *Computing in Science & Engineering, 13*(2), 22-30. https://doi.org/10.1109/MCSE.2011.37

Wickham, H. (2016). *ggplot2: Elegant graphics for data analysis*. Springer-Verlag. https://ggplot2.tidyverse.org

Acknowledgments

I would like to acknowledge the use of the following libraries and frameworks that were instrumental in the development and evaluation of the model:

- **TensorFlow**: For providing the deep learning framework used to build and train the hybrid LSTM-CNN model.

- **Keras**: For offering the high-level API built on TensorFlow, which facilitated the design and training of the neural network architecture.
- **Pretty_MIDI**: For the MIDI data processing that allowed the extraction of pitch sequences from the musical compositions.
- **NumPy**: For efficient numerical computation and array manipulation, which was crucial in handling the data preprocessing and feature extraction.
- **Scikit-learn**: For providing the tools necessary for data splitting, evaluation metrics, and other essential machine learning tasks.