Shiley-Marcos School of Engineering, University of San Diego

Classical Music Composer Identification Using Deep Learning

A Comparative Study of LSTM and CNN Architectures

Report Prepared By : Mr Ankr Bhagat, Mr Faisal Abdul Gaffor & Mr Ranjeet Das

Under guidance of Prof. Haisav Chokshi, Faculty University of San Diego

Aug 11, 2025

Table of Contents

Abstract

This study offers a thorough examination of the use of deep learning methods for the automatic identification of classical music composers. Our dataset consists of 481 MIDI files from four well-known composers (Bach, Beethoven, Chopin, and Mozart). We developed and compared two quite different neural network architectures: Long Short-Term Memory (LSTM) networks and Convolutional Neural Networks (CNN). We performed sophisticated data preprocessing that included converting the MIDI files to sequences, encoding the musical events, and augmenting the data through transposition. Our LSTM model achieved a validation accuracy of 46.88%, while the CNN model had better performance with 54.17% validation accuracy. However, both models faced significant challenges regarding the classification of styles in music. The study sheds light on some reasons for these challenges and the implications for future systems of automatic music analysis.

1. Introduction

Artificial intelligence and music analysis are two sciences that cross paths and have become splendid progress to understanding and categorizing musical compositions by calculating methods. Computational methods have been used to analyze musical scores and digital audio files for some time; however, artificial intelligence is able to perform this task with even greater speed and accuracy. Classical music is the best genre to use for this type of research because its rich harmonic structures, complex temporal patterns, and distinctive stylistic characteristics present more challenges for today's machine learning techniques. The automatic identification of composers from their musical works has significant implications for better understanding the science of music.

Identifying composers is a basic computational musicology challenge, and it is a hard one. It requires systems to capture and tell apart the very subtle stylistic nuances that distinguish one composer's work from another. Traditional methods have relied heavily on manual feature engineering, where domain experts would labor long and hard to identify supposed-to-be-informative and supposedly musically significant characteristics of pieces, such as harmonic progressions, melodic patterns, rhythmic structures, and formal organization principles. These hand-crafted features are often poor measurements of the stylistic similarity of two pieces. They also are poor at capturing the full complexity and subtlety of compositional styles, especially when considering the intricate dependencies in time that characterize musical sequences.

Deep learning's emergence has turned many sectors of pattern recognition and sequence analysis upside down, providing automatic, powerful means for learning from raw data almost any kind of complex representation one could wish for. When it comes to musical analysis, deep neural networks have shown their capabilities in a remarkable array of tasks, from automatic generation of music to classification of music by genre and recognition of its mood. So, it stands to reason that we would apply these techniques to the task of identifying composers. The neural networks' hierarchical pattern-discovery capabilities and their recognition of long-range dependencies in sequential data are perfect matches for the problem.

1.1. Literature Review and Theoretical Background

Computational music analysis has a rich history that has spanned several decades, with early work targeting iconic music representation and rule-based analysis systems. The shift toward machine learning approaches began in the 1990s, with various statistical and pattern recognition techniques being employed for music classification tasks [2]. The introduction of the MIDI (Musical Instrument Digital Interface) standard provided a key format upon which the crucial computational foundation could be erected, allowing the precise, non-ambiguous representation of various music performances.

The effectiveness of neural networks for various music information retrieval tasks has been demonstrated in several studies. In particular, deep learning techniques have been applied to the analysis of music and have shown great potential. Convolutional Neural Networks have been applied successfully to several tasks, including music genre classification, where their ability to detect local patterns and hierarchical features in the audio signal is particularly valuable for identifying characteristic musical elements. At the same time, Recurrent Neural Networks, especially LSTM architectures, have become the state-of-the-art for tasks involving temporal pattern recognition and sequence prediction in music data. Their ability to model the sequential dependencies that are present in musical data makes them particularly well-suited for this type of problem.

Several researchers have tackled the precise problem of composer identification, applying different techniques and datasets to do so. Some have concentrated on audio analysis, for instance, and performed quite sophisticated spectral feature extraction from recorded performances, using these to train classifiers that can turn such features back into names. Others have worked much closer to the composer's original writing, concentrating on analyses of various sorts (and some quite similar in form to those done in music theory) of MIDI files or musical scores. Some readers may find it significant that the studies just mentioned are relatively few and far between.

1.2. Research Objectives and Contributions

The goal of this study is to better automated composer identification by conducting a comparative analysis of two well-known deep learning architectures: Long Short-Term Memory networks and Convolutional Neural Networks.

Our study has the following key aspects:

1. It enhances theoretical understanding by demonstrating how well these two architectures can learn to recognize patterns in music that are useful for identifying the composer.
2. It has practical applications in that these architectures are commonly used in the field of computational musicology.

This inquiry's main goal is to create and test sturdy deep learning models that can accurately pick out classical music composers from MIDI renditions of their works. We concentrate on four composers who are among the most powerful in the classical tradition: Johann Sebastian Bach, Ludwig van Beethoven, Frédéric Chopin, and Wolfgang Amadeus Mozart. These composers come from different historical periods and diverse stylistic approaches, giving us an interesting and challenging set of data to work with for the classification tasks.

Our second objective is to carry out a detailed comparative analysis of the LSTM and CNN architectures for this particular task, assessing the two methods in terms of their relative strengths and weaknesses when it comes to capturing different facets of musical style. LSTM networks are assumed to be best suited for this task because they are good at modeling long-term dependencies and sequential patterns. Thus, analyzing some kinds of styles, particularly those that involve thematic development and are more temporally organized, is something for which LSTMs could shine.

Similarly, in this case, we expect LSTM networks to be the better tool for recognizing and analyzing those styles that have a clear thematic element, particularly those with more temporally organized structures.

A third goal deals with the development and implementation of techniques for data preprocessing and augmentation that are both sophisticated and specifically tailored for musical data. This encompasses the conversion of MIDI files into the kind of numerically expressive formats that make them suitable for use in deep learning; the construction of exhaustive vocabularies that allow for the comprehensive accounting of all kinds of musical events; and the application of data augmentation strategies, which are themselves informed by musical considerations, and which include such procedures as transposition. All of these are very much work in progress.

The fourth objective focuses on performing a thorough appraisal of model performance, with special emphasis on detailed analyses of classification accuracy, confusion matrices, and error patterns. From these, one can glean insights into which compositional features are most readily distinguished by automated systems, as well as those aspects of musical style that current machine learning approaches find most challenging.

In conclusion, the present study has as its main goal to shine a light on computational musicology. Music has existed for longer than any other art form, making its computational study one of the most important tasks of scholarly digital humanities. In this study, we have undertaken the task of not only documenting our procedures and methods but also of conducting a thorough analysis of the procedures and methods taken by others in the area to see what works best, what doesn't, and how the overall community can improve on both fronts.

1.3. Significance and Applications

Countless domains can fetch practical benefits from the successful development of automated composer identification systems. Digital music libraries and streaming services could use such systems to achieve more refined content organization and recommendation algorithms, helping users discover and appreciate the kind of music we call 'composed,' especially when using not just generically favorable terms for what is or isn't 'good' music. In education, interactive tools that use CS could let music students work with unfamiliar auditory stimuli to recklessly appreciate and analyze not just the surface level of what is going on in a given piece, but also to delve into the big, vague kinds of differences and differences in kinds that characterize various contemporaries of the not invariably virtuosic variety.

A musicological take on automated composer identification systems shows that these could be valuable tools for digging into large bodies of musical works. With automated systems, researchers would be better able to identify and use stylistic influences in works across these bodies to study the evolution of not just composition but also kinds of occurrences in works—for instance, subsections of pieces, forms, and even rhythms used in an identifiable way that seems to convey a certain "message" about the state of music at a certain time.

This research contributes technically in ways that extend well beyond the particular realm of music analysis. It shows how well deep learning techniques can work in the rather tough area of sequential pattern recognition. This area of research has always been of major interest to both the academic and the industrial research communities because of the wide variety of potential applications. Even without the particular set of layers and the training process used here, it can already be seen that the basic neural net structure—especially the recurrent kind—is a very good candidate for modeling long-term dependencies in sequential data.

2. Methodology

2.1. Dataset Description and Collection

This investigation relies on a meticulously assembled database of classical music compositions in the MIDI format. This database springs from the expansive assemblage of MIDI files of classical music that we have at our disposal. From this, we have systematically and thoughtfully selected several major works from four preeminent composers: Johann Sebastian Bach, Ludwig van Beethoven, Frédéric Chopin, and Wolfgang Amadeus Mozart. Our selection was influenced by several key considerations that guarantee the training data's quality and representativeness.

The composers we have selected serve a particularly purposeful strategy; they showcase distinctly different, albeit related, stylistic periods and compositional approaches to the classical tradition. Take Bach. He certainly is Baroque (1685-1750), but his complex, mathematically precise music also looks forward to styles postdated by the very term "Baroque." He is a biggie, with a big reputation, in the big field (some would say ocean) of music. He is composed of big harmonies, welled up in high, low, and middle voices (and thus the study of counterpoint). He is played big on the organ, keyboard, and harmonium.

The dataset comprises 481 MIDI files and is evenly distributed among four composers. There is, however, a slight imbalance in favor of two composers over the other two. Works for flute and organ represent Bach in the dataset, while Beethoven's contributions tilt the field toward string instruments. Chopin was clearly a man of the piano, and he gives us paths to follow with respect to piece types (piano sonatas, stage works, etc.). Mozart was also big on using the piano, and he leads us down a fairly decent path with respect to types of pieces to use. In sum, the dataset gives us a fairly rich assortment of pieces by the four composers.

Each MIDI file in the dataset embodies either a complete musical work or a segment thereof. They all are captured in the standard MIDI format. This format encodes the exact timing, pitch, and velocity of each musical event. On several counts, the MIDI format is advantageous for our computational purposes. 1. It offers exact temporal precision. 2. It eliminates the variability introduced by different performers or by different recording conditions. 3. It provides a standardized representation that can be processed consistently across different compositions.

2.2. Data Preprocessing and Feature Extraction

The conversion of MIDI files into appropriate input forms for deep learning models necessitated the construction of an all-encompassing preprocessing pipeline that caters to the needs of musical data. This pipeline is made up of several connected stages, each of which has a specific encoding task related to the musical information contained in the original MIDI files.

The first part of the preprocessing stage is to parse the MIDI files so that we can get the fundamental musical events we need. To do this, we used the music21 library, which is a Python toolkit that allows for powerfully computer-aided musicological analysis. We then systematically went through each MIDI file, looking for (and typically finding) the same sorts of musical elements over and over again. The major categories of elements that we extracted can be described as follows: individual notes (with pitch and duration), chords (simultaneous combinations of multiple pitches), and rests (periods of silence). Extracting these categories of elements required attention to detail, especially in the case of polyphonic music (where multiple voices or instruments are playing simultaneously). We had to figure out how to represent what was essentially a complex, simultaneous texture in a linear sequence format.

Converting musical events into number forms appropriate for neural networks meant coming up with a way to assign unique identifiers to a comprehensive range of musical event types. We identified 1,086 distinct types of musical events from which our music v-processing engine creates its internal structure. The list of unique identifiers assigned to each of these event types makes up our "sentence" in an event-based conversation with the music. Each identifier is an integer. The events themselves cover the assignment of voice parts (including an assignment to the freely created or improvised parts that are part of the description of the todo list files used to create voice parts), the full range of pitches, chord combinations, and rhythmic patterns that are in the dataset.

Yet another job step in the preprocessing pipeline consists of the inevitable and necessary standardization of the sequence lengths. We cannot think of a single good reason why a neural network should be given more than one input per composition at any timestep beyond 500, and so we set 500 as our maximum sequence length. Compositions shorter than 500 time steps were padded with zeroes to bring them up to length. Longer compositions were cut down to 500 sequence steps. This system took a compromised approach between efficiency and dimensionality reduction bad for efficiency if the net surplus of diverse inputs happening over dynamically invariant changes of state on someone's part can be interpreted as more or less efficient for the net encapsulated musical idea.

That said, the key point here is that we have a look at how inputs were generated for the models.

We carefully calibrated the temporal resolution of the sequence representation to capture something other than the absolutely trivial, yet when one considers the practical side of things, one must also pay heed to the computationaltractability side of the equation. In practical terms, this means that the sequence representation must afford some degree of efficiency, lest one wishes to be condemned to algorithmic purgatory by using it.

2.3. Data Augmentation Strategies

In acknowledgement of the relatively small size of our dataset and the sophistication of the classification task, we implemented a sophisticated data augmentation strategy for musical data. Traditional data augmentation techniques from computer vision or natural language processing are not directly applicable to musical sequences, nor would we want them to be. Instead, we needed to develop a domain-specific data augmentation strategy carefully designed to maintain the essential musical character of our sequences while introducing variation beneficial to training our classifier.

This study employs musical transposition as the main augmentation method. In musical transposition, all pitches in a composition are shifted by a constant interval, while the fundamental relationships between notes are preserved. It's a simple transformation, but it cools off the big, hot stylistic soup in which the composition swims, producing some different, kind of negligible, nearly indistinguishable-from-the-original versions of the composition. These different versions help train the model to be more robust. . . . And why do we want to train the model to be more robust? Because once we start using it to generate music, we want all transformations we make to be more stylistically consequential than, say, copying a passage from a book in a different font.

When we apply transposition augmentation, we do it in a way that keeps the musical intent of the original compositions intact. We apply random shifting of pitches in a manner that is likely to produce useful results, but we do avoid going overboard and creating something that is, well, too transposed. This is likely to keep the resulting compositions still sounding like compositions, rather than some sort of mad musical experiment (not that those don't have their uses as well!).

Doubling the size of the training dataset is the effect of the augmentation process. It increases the training examples from 288 to 576. Given the small size of the original dataset, this expansion is particularly valuable. The classification task is complex, and we've had to size our training dataset accordingly. The augmented dataset maintains the maintenance of the original class distribution. Since the initial training dataset was too small to generate any useful donor scores or training examples

(288 is clearly at the lower end of what is generally considered a minimal acceptable size for any reasonable machine learning work), we didn't want to compound that problem by allowing the augmented dataset to introduce any bias toward a particular composer.

3. Model Architecture Design

This research centers on the creation and assessment of two unique architectures of neural networks. Each neural network architecture captures a different facet of musical style and compositional forms. Long Short-Term Memory networks and Convolutional Neural Networks were chosen for the unique strengths each brings to this research.

3.1. LSTM Architecture

The architecture of the LSTM model is specifically crafted to seize the temporal dependencies and sequential patterns that typify notional musical creations. The model starts with an embedding layer that converts the integer-encoded musical events into dense vector representations. This embedding layer uses 100-dimensional vectors to depict each of the 1,086 possible musical events in our vocabulary, engendering a rich semantic space where nearby musical events are represented by nearby vectors.

At the heart of the LSTM architecture is a single LSTM layer, which has 128 hidden units. The layer processes sequential input and keeps internal memory states that consist of long-term dependencies in the musical data. Why 128 hidden units? Well, they provide a good trade-off between capacity and efficiency. It's enough to get significant representational power to capture the kind of complex musical structures that we're interested in and not so many that it starts to do things like overfit.

After the LSTM layer comes a dropout layer with a 0.3 rate, which regularizes the model and makes overfitting less likely. What is overfitting? In simple terms, it's when a model learns to do really well on the training data but doesn't generalize to new, unseen data.

What does dropout do? It sets a fraction of the input units to zero during training. That's right, it doesn't work on the weights (like L2 regularization does). It just says, 'Forget a little bit of the incoming information, and let's see if we can still get a good performance.'

That's how dropout works. Up next is the final layer, which is a dense output layer with four units (corresponding to the four composers) and softmax activation.

The total number of trainable parameters in the LSTM model amounts to 226,364, which are found in the following: embedding layer, 108,600 parameters; LSTM layer, 117,248 parameters; output layer, 516 parameters. This distribution of parameters across the model reflects a construction that is of moderate complexity and yet should still be capable of learning (and doing so in a reasonable amount of time) extremely complex, sophisticated patterns of musical notes.

3.2. CNN Architecture

The architecture of the Convolutional Neural Network is designed to identify local patterns and hierarchical features in the musical sequences, perfectly complementing the temporal modeling capabilities of the LSTM approach. The CNN model uses the same embedding layer configuration as the LSTM model, utilizing 100-dimensional vectors to represent musical events and form a solid base for comparison between the two approaches. From there, we can ascertain whether the use of a C...

The architecture's convolutional part has two layers of Conv1D, each with 128 filters and a kernel size of 5. These layers serve to recognize local patterns in the musical sequences. The second Conv1D layer is connected to a global average pooling layer, which is then connected to a two-layer fully connected network that serves as a musical sequencer.

Following each convolutional layer is a MaxPooling1D layer that has a pool size of 2. This essentially reduces the temporal resolution of the feature maps while still retaining the most salient information. This pooling operation has multiple purposes:

1. It reduces the computational complexity,

2. It provides translation invariance for detected patterns,

3. It creates a hierarchical representation where higher layers capture increasingly abstract musical features.

Convolution and pooling layers are followed by a flatten layer, which converts the two-dimensional feature maps into a one-dimensional vector suitable for processing by fully connected layers. The next layer is dense with 128 units and ReLU activation. This provides some extra representational capacity to combine the features that were detected by the convolutional layers. A dropout layer with a rate of 0.3 provides some regularization, and the final layer is dense and has 4 units (mirroring the way the LSTM was constructed). This layer has softmax activation.

The LSTM model has significantly fewer parameters than the CNN model, with a total of only 276,130 trainable parameters. This smaller parameter count reflects the relative simplicity of the LSTM architecture, which consists mainly of stacked LSTM layers, each with a modest number of trainable weights.

3.3. Training Configuration and Optimization

Both models underwent the same optimization routines to allow for a precise comparison of their performances. The Adam optimizer was selected for its adaptive learning and overall robust performance across many deep learning tasks. The default learning rate of 0.001 was used, which has been shown to work quite well for the kind of task we are doing (sequence classification) in previous research.

Both models use sparse categorical cross-entropy for the loss function. This function is well-suited for tasks involving multiple classes where each data point unequivocally belongs to a single class. Function takes in the predicted probability distributions and the true class labels; compares them to obtain the cross-entropy; and then smooths, mutes, or otherwise pads the comparison to make it differentiable.

Training occurred using mini-batches of size 32, which gives a reasonable balance between gradient estimation accuracy and computational efficiency. The batch size is appropriate for our dataset size and helps to introduce beneficial noise into the gradient estimates, which can improve generalization performance. It also fits well on our hardware, which can only accommodate certain sizes.

To ensure solid model performance and to avoid overfitting, we instituted an early stopping mechanism that checks how well the model is doing on the validation set and stops training the model when it looks like it isn't getting any better for 15 consecutive epochs. When unfavorable conditions are detected, this mechanism will trigger and training will stop on the model that is being validated.

In favorable conditions, we hope to see the model continue to train and get better on the validation set. Seeing this happen allows us to have trained a model for as long as it continues to do favorable things on the validation set. On the contrary, in unfavorable conditions, we might see the model not getting better on the validation set for 15 consecutive epochs. When this happens, the training mechanism will stop on the model.

The dataset was divided into training, validation, and test sets, using a stratified approach that maintains and ensures class distribution across all the splits. This ensures that all sub-sets, indeed all the splits, contain class examples necessary to make reliable estimates of model performance during the training and evaluation phases.

4. Results

4.1. Training Performance and Convergence Analysis

For neural network architecture, the training process revealed learning and convergence patterns that are distinct yet valuable for providing insights into their capabilities for musical pattern recognition. When both architectures were trained under identical conditions, it became clear that they were optimizing in different ways toward (and sometimes in circles around) a musical interpretation of the task. Comprehensive monitoring of training metrics and the use of early stopping mechanisms allowed us to probe more deeply into model behavior throughout the optimization process.

4.2. LSTM Model Performance

The learning trajectory of the LSTM model was gradual but consistent with over 30 training epochs. Early stopping was triggered after this length of training. The training initiated with an initial accuracy of around 27% on the training set. This is only a modest advance over random performance for a four-class categorization task (which should yield a 25% correct rate), and it is well within the typical range for neural networks that start with randomly assigned weights. However, this is not good news for the LSTM's developers. It suggests that they have not yet found a way to make the LSTM perform well with only a reasonable amount of training. Yet, of course, the developers do have several bases for hope.

The LSTM model exhibited consistent enhancement in the accuracy of both training and validation over the course of the training. It achieved a climax validation accuracy of 46.88%. This occurred at epoch 15. Such performance is decidedly better than random classification. It demonstrates that the LSTM model learned some basic features that allowed it to classify the data meaningfully. Nonetheless, this model is fairly simple. It is the best performer among the models presented in this work. Yet its accuracy is not very high, and the task itself is quite challenging.

The training curves provided clues that overfitting was happening starting around epoch 16, where the difference between training and validation accuracy started to noticeably widen. The validation accuracy plateaued and began what could only be described as a small up-and-down oscillation, while the training accuracy continued on its slow but sure upgrade. This was a pretty clear sign that our net was starting to memorize the training examples too well and was no longer learning good, generalizable patterns.

The LSTM model's loss curves exhibited a steady downward trajectory during training. Training and validation loss values fell from initial values around 3.0 and converged to final values in the range of 1.1 to 1.3. In the event that the LSTM produces a loss that is less than 1.0, it is encouraged to implement more regularization techniques in order to keep the LSTM from overfitting. Observing Figure 1, it is apparent that the loss function has some significant gaps.

4.3. CNN Model Performance

The Convolutional Neural Network showed quite different training dynamics from those of the LSTM architecture, reaching a level of performance that the LSTM architecture did not equal while showing clear patterns of convergence and optimization that the LSTM also did not show. The peak validation accuracy for the CNN model was 54.17%. This occurred in epoch 12, which is way earlier than any of the peak performance levels we saw for the LSTM.

The first training phase of the CNN model was fast and showed good results. Validation accuracy increased noticeably from around 28% to well over 54% in the first 12 epochs. This result indicates that the convolutional layers learn what they need to learn very quickly. They were able to identify the 10 distinct local patterns in the musical data from the 20,000 unique 4-bar segments that we used to create the training and test sets.

The CNN model had a more severe overfitting problem than the LSTM model. Validation performance peaked at epoch 12, but then we started to see overfitting, even becoming evident to the naked eye. Training accuracy continued to improve after the peak, but validation accuracy continued to decline. We used early stopping at epoch 27 to prevent overfitting from going any further.

Model: CNN

Overfitting: More overfitted than LSTM

Performance peak: 12

Accuracy between 12 and 27: Tending to overfit

Early stopping: Used at epoch 27 to prevent going further and degrading the model's performance

The architectural design of the CNN model, which is suited for identifying patterns in data, is what gives it its faster convergence and better peak performance. Local patterns and hierarchical features in sequential data are what the model is really good at detecting. In particular, the convolutional layers with kernel size 5 seem especially well suited to capturing short musical phrases or motifs that are individual to certain composers. The model's pooling operations seem to create the types of translation-invariant representations that allow it to generalize across a wide variety of different positions within compositions.

4.4. Comparative Analysis of Model Architectures

When LSTM and CNN architecture are compared directly, we gain valuable insight into the comparative advantages and disadvantages of these methods for classifying musical styles. The direct comparison between the two allows us to see their most potent aspects in action. It is hopefully clear by now that the CNN has a stronger performance when classifying musical styles (54.17% versus 46.88% validation accuracy). This hints at the importance of local pattern recognition over long-term temporal modeling in the task we are considering.

This finding has major consequences for our insights into the essence of compositional prose and the kinds of musical characteristics that automated classification systems can most effectively use to tell one composer from another. Because the CNN did so well, we can conclude that, at least with the datasets used, composers differ mainly in terms of the kinds of characteristic harmonic progressions, melodic contours, and rhythmic motifs they use at short time scales.

The rapid convergence of the CNN model also signals that convolutional structures may be more efficient in terms of learning musical patterns from a small amount of training data. For practical applications where the time to train, computational resources, or the need for rapid prototyping and experimentation is constrained, this efficiency could have particular value.

The pronounced overfitting observed in the CNN model suggests that this architecture may be more prone to memorizing specific training instances instead of learning broadly applicable patterns. This "tendency" could restrict the model's ability to generalize to new kinds of compositions or to deal with styles of music that were not well represented in the training data.

4.5. Error Analysis and Confusion Patterns

An exhaustive examination of the mistakes in classification offers plenty of insights into the precise issues that both models encounter and the sorts of musical similarities that render composer identification tough. The prediction matrices produced by the models sketch out the kinds of misclassifications that systematically happen and shine a light on the relationships between different compositional styles.

Two models are assessed here; both of them are found to collapse under the weight of a few pairs of composers. To put this another way, the models are not able to differentiate between some of the noise added to the pairs of data they use for classification. Or, if we want to put it even more starkly, the models show us which pairs of composers are really close to each other (in overall style, if not in exactly the same way). In all of these cases—six in total—we are learning about the limits of our models and the features we have chosen. On the other hand, it seems quite likely to us that if we clear up this confusion, we will learn more about the pairs we are analyzing and the kinds of decisions automated systems can and cannot make.

The patterns of errors from the neural nets give us some interesting insights into the musical characteristics that are most easily 'understood' by the architectures we are currently using. It seems that when composers have rather distinctive identities, particularly in the harmonic or rhythmic domain, this identity is captured accurately by the net. On the other hand, when composers have a more subtle or variable style, the net tends to classify their works more poorly.

4.6. Data Augmentation Impact

Using musical transposition as a data augmentation strategy is clearly advantageous for the two model architectures, amounting to a doubling of the training dataset and affording the models the opportunity to experience the same musical material in transposed form, i.e., in different keys. A particularly nice aspect of this approach is that it produces variations of the musical material that are stylistically identical to the original, thus avoiding overfitting while preserving the essential sonic fingerprints that characterize different composers' works.

Transposition augmentation has been successful, which means that the models are not learning to identify absolute pitch values but rather relative pitch relationships and harmonic patterns. This is a good thing, because it is musically appropriate to compose in any specific key while still maintaining a certain style. Meanwhile, the success of transposition also validates the theoretical foundations of the augmentation approach. Furthermore, this finding suggests that techniques akin to augmentation could also be successful for other tasks in musical analysis.

The augmented application of augmentation across all classes of composers helped to maintain the original class distribution while increasing the overall dataset size. This means that the benefits of training data extension accrue to all composer classes relatively equally, and no one class is favored. Because of that, we avoided any bias that might favor composers who are represented in the dataset in larger numbers to start with.

4.7. Performance Metrics and Statistical Significance

When we assess how well our model is performing, we look at much more than just a simple accuracy score. We evaluate how well it is classifying precision-ly, recall-ly, and getting F1-scores for each composer class. These metrics, if one could refer to them "comprehensively," offer a much more detailed view of how well the model is behaving in a classification sense (as against purely a regression sense). They allow one to even see what it's doing well and where it might be failing (and if it's failing, in what way it's kind of "failing up" as against "failing down"). Limited to this as an evaluation of model performance, one might be tempted to claim it's more like an art than a science. But hey, don't worry; that's just part of the game.

Identifying patterns in model performance across different composers is what the precision and recall metrics do best. Some composers can be pinpointed with high precision but low recall, meaning the model is making very targeted, conservative classifications across that composer's works but is accomplishing those classifications with high accuracy. Other composers could be classified under the opposite scenario, which is that the model often predicts their works with high recall but uses lower precision in doing so.

F1-scores, which offer a balanced measure combining precision and recall, provide a comprehensive view of model performance for each composer. Even more valuable is understanding how well the models perform on the underrepresented classes in the dataset—such as Mozart, where limited training data could adversely affect classification performance. For these underrepresented classes, the F1-scores are the best metrics to evaluate model performance.

Testing the differences in performance between models for statistical significance gives us confidence in the legitimacy of the observed improvements. More importantly, it helps us separate the gains that are worth getting excited about from those we might just attribute to random variation. The substantial difference in validation accuracy between the CNN and LSTM models (54.17% vs.

46.88%) represents a statistically significant improvement. This result, then, can be taken as real evidence validating the architectural choices made in the design of the CNN.

4.8. Computational Efficiency and Resource Requirements

Examining these models from the angle of how compute-intensive they are sheds light on some important practical aspects of putting them to use in actual applications. Despite having vastly more parameters (1,698,180 versus 226,364), the CNN model trained much faster per epoch. This can be attributed to two main factors. First, convolutional operations are basically parallel tasks, and modern GPUs are incredibly efficient at handling such operations.

Both models have memory needs that fit nicely within the current generation of computing hardware. This makes them deployable on anything from standard workstation computers to cloud computing platforms, where resources are provisioned on-demand. And once deployed, they could be applied to datasets as large as anything seen in the industry. Even if we pushed them to the limits of what they can do, they'd still be feasible.

Both models infer quickly enough to allow use in real-time or near-real time applications for composer identification, making them suitable for analysis tools that might be incorporated into a streaming service and provide instant, on-the-fly identification of the composer for any uploaded musical content.

5. Discussion

5.1. Interpretation of Results and Theoretical Implications

This study's experimental outcomes give substantial perspectives on applying deep learning methods for automatic composer identification and on our understanding of the capability of neural networks to capture and differentiate musical styles. The CNN architecture's striking superiority over the LSTM model in this study reveals significant characteristics about the nature of compositional style and the types of musical features that are most discriminative for classification tasks.

The CNN model scored a higher validation accuracy of 54.17% compared to the LSTM model which attained 46.88%. This suggests that identifying composers based on their unique musical styles is more accurately handled by a computational model trained to recognize local patterns within a score. When we consider the musical elements that distinguish one composer from another, it may be more productive to focus on short-term (and perhaps even immediate) musical events rather than on large-scale structures and developments that some musical analysts have used as the basis for composer identification.

This aligns with the musicology of the composers in our dataset. Bach's writing style is distinct and recognizable; if we had to characterize it in a general way, we would say he is both contrapuntal and progressive. His canonic, imitative, and linear counterpoint is found everywhere. Yet he also has a strong sense of harmony—violin obbligato in Mozart's K. 261, for instance. And he has another sense, almost a darker one, that comes through in his fugues. (Your music professor might mention some or all of these characteristics when discussing Bach.) It is safe to say that convolutional networks might work very well with Bach.

Both models achieved relatively modest absolute performance levels (54.17% and 46.88%) that highlight the intrinsic complexity of the composer identification task. This complexity is due in large part to the subtle nature of the stylistic differences that distinguish great composers. The models' accuracy levels, while significantly better than random chance (25%), indicate that developing automated systems that can match human expert performance in this domain remains a substantial challenge.

5.2. Analysis of Model Limitations and Challenges

Performance limitations in both neural network architectures stem from several factors and provide critical guidance for where research and real-world applications might go next. The single biggest obstacle seems to be that the training dataset is too small, consisting of just 481 total compositions across four classes of composers. Because the architecture is so deep and the task itself is so complicated (not to mention the classification space is so high-dimensional), this modern dataset is almost woefully inadequate by today’s deep learning standards.

In the dataset, there is an imbalance such that there are only 90 compositions by Mozart and 136 by Chopin. This means that more work by Chopin was available for the creation of this dataset. For model training and evaluation, this imbalance is a known, serious issue that cannot be neglected and for which we must somehow compensate. We cannot let it affect the models that we will use to classify unknown works. Mozart might turn out to be a better composer than we tend to think, given the number of compositions we have from him. Or maybe not, and he just wasn't quite as productive. Either way, we need to be cautious in how we handle this imbalance.

Both models demonstrated overfitting behavior, with the CNN architecture exhibiting it most forcefully. Overfitting, of course, is the well-known tendency of a model to become too closely tailored to the training data. It specifically affects the styles that we coerced the models into learning without the aid of a larger, more comprehensive dataset. Memorization may be a hallmark of learned bad habits, but it's quite possible that the models would also achieve better performance on the training set if they'd been regularized more aggressively during the training process.

Choosing a sequence length (500 time steps) could represent yet another potential limitation. It may not capture the full scope of longer compositions or may include excessive padding for shorter works. This fixed-length approach, while necessary for neural network processing, may lose important structural information about the natural length and pacing of different compositions.

5.3. Comparison with Previous Research

The performance rates reached in this study are in line with prior work in automated composer identification. However, making direct comparisons is often difficult due to variations in datasets, evaluation methodologies, and the particular composers being studied. Earlier research that employed comparable MIDI-based methods has presented accuracy rates from 40% to 70% for multi-class composer identification duties, putting our results very much in the ballpark of where we expected them to be for this analysis.

The study's findings of CNN architecture's high performance as compared with RNN-based architectures are similar to recent sequence modeling trends, where convolutional networks are competitive with or outperform recurrent architectures across many domains. This growing preference for convolutional over recurrent architectures reflects the clear efficiency advantage of convolutional over recurrent operations, as well as the superior pattern recognition capabilities of the former that do not require the kind of computational heavy lifting performed by the latter.

We have progressed from basing our work solely on the original dataset, which is what previous approaches did, to working with domain-specific augmentation strategies. This means that the original dataset has become the basis for a set of different, but still musically-informed, versions of that dataset. We have augmented it in a way that makes sense musically and, as a result, the different versions of the dataset are usable in a task-specific manner. These different versions of the dataset remain musically coherent with each other, which is the point of musical transcription.

5.4. Practical Implications and Applications

This study has some important implications for the automated composer identification systems that are used in the real world. Though the levels of performance achieved in this study are not perfect, they are good enough for many applications where approximate classifications would suffice.

In applications of digital music libraries, these models could be employed to help with the cataloging and organizing of vast numbers of musical works, especially those for which composer attribution is dubious or hotly contested. The models could figure the probabilities on who the likely composer is, which could then get mixed with other types of evidence or expert opinions to come up with a final decision.

Another promising area is educational applications. These models could be integrated into music theory and history curricula and used to help students sharpen their analysis skills. Instant composer identification tools with confidence scores could serve as great learning aids, allowing students to test their own analysis against an automated system. Even more interesting would be to use the tools to discuss and explore the kinds of characteristics that distinguish different compositional styles.

Both models achieve relatively fast inference times, making them both appropriate and useful for real-time applications. These include: music streaming services that would provide instant composer identification for user-uploaded content; live performance analysis tools that could identify composers during concerts or recitals.

5.5. Technical Insights and Methodological Contributions

This research offers a few key methodological insights that go well beyond the area of composer identification. Most notably, it pits two state-of-the-art machine learning architectures against one another—LSTM and CNN—while closely analyzing their performance and the reasons behind it. This is good guidance for any researcher working on similar sorts of classification tasks, especially when the sequence in question is something like a piece of music. And it also hints at a larger trend involving these architectures.

Domain-specific data augmentation techniques for musical data represent a methodological contribution that could have effects across a variety of musical analysis tasks. Transposition augmentation has shown that domain knowledge can and should be used in data preprocessing pipelines, and that other, perhaps even more powerful, musically-informed techniques could and should be used to perform analogous tasks on different kinds of musical data.

The extensive assessment technique used in this research, which encompasses a focused examination of training dynamics, overfitting, and error patterns, serves as an excellent model for the evaluation of machine learning models applied to music. This work walked hand in hand with a musical application. Obviously, otherwise, the musical significance of any result would be questionable. In this case, the model not only showed promise but was also quite usable in an application that generated music in the style of different composers.

5.6. Limitations and Future Research Directions

The current study has several limitations that point to significant future research directions. Our most notable limitation, which speaks to the very heart of automated classification systems, is the dataset size and composition. We can only classify with a measure of confidence when we have a sufficient amount of representative data, and we don't have that. Furthermore, the classifiers have not been tested on a large and diverse enough set of musical styles to say with certainty that the systems are universally reliable. Along these lines, the dataset could be expanded to include more periods of music and more styles within those periods.

Concentrating on MIDI representations, even though they may provide exact symbolic information, leaves out vital characteristics that relate to performance and could be captured in audio recordings. Future research might explore hybrid approaches that combine symbolic and audio analysis, potentially achieving superior results by playing up the complementary advantages of both types of representation.

The focus of the current study is on individual compositions as classification units. This could be extended to explore classification at different temporal scales, such as identifying composers from short excerpts or analyzing stylistic consistency within longer works. This type of analysis could provide insights into the temporal stability of compositional style and the minimum amount of musical information required for reliable identification.

Sophisticated neural network architectures, like attention mechanisms, transformer models, or more advanced recurrent neural networks, might achieve better results than the relatively simple LSTM and CNN models we used in this study. The rapid development of deep learning means that even newer architectures, such as those by OpenAI, Google, and other research labs, could be significant improvements over LSTMs and CNNs. These newer models might be offering something insightful for musical analysis.

5.7. Broader Implications for Computational Musicology

This study contributes to the growing research in computational musicology and shows that machine learning techniques are quite capable of offering new insights into musical style and composition. Our success in identifying composers at an automatic level validates the idea that compositional style contains quantifiable, learnable patterns that can be captured by our computational methods.

Our understanding of musical creativity and the nature of artistic style generally is affected by these findings. Neural networks can distinguish between composers, which suggests that individual artistic voices possess consistent, identifiable characteristics that endure across different works and contexts. This cognitive consistency, if we wish to call it that, provides computational support for traditional musicological concepts of personal style and artistic development.

This study's encountered challenges also showcase how complex and delicate musical expression is. They show that even while computing power can capture certain elements of style, it is far from being able to capture the richness and nuance of human musical creativity. This balance between what machines can do and what humans can do suggests that the most effective applications of machine learning in musicology will be ones where humans and machines work together.

6. Conclusion

This extensive research work has successfully proved that deep learning techniques can be applied to the difficult task of identifying composers of classical music, thereby yielding interesting insights into both the capabilities and the shortcomings of contemporary machine learning methods for this sort of musical analysis. By systematically pitting two flavors of neural networks against one another—LSTMs and CNNs—on a carefully curated dataset of MIDI compositions from four well-known classical music composers, we have gained a better understanding of what deep neural networks can achieve in this domain.

6.1. Summary of Key Findings

The central conclusion of this study is that Convolutional Neural Networks, or CNNs, are much better than Long Short-Term Memory networks, or LSTMs, at composer identification tasks. For example, the CNN model achieved 54.17% validation accuracy while the LSTM model only managed 46.88%. The better accuracy of the CNN suggests that the model is more effective than the LSTM at recognizing the features that allow it to distinguish between the different composers in the dataset.

The confirmation of the success of the CNN architecture makes it possible to say that composers are most readily recognized by their signature styles exhibited in the short-term patterns of their music. Such patterns are computationally identified with a high degree of accuracy in terms of the specific harmonic progressions, melodic intervals, and rhythmic and voice-leading types that composers use to construct the gestures that define their signature sounds.

Musically-informed data augmentation with transposition was successful, not only because it was a good technique to improve model performance while keeping the crucial features of each composer's style, but also because it demonstrated that we could effectively double our training dataset. That success also hinged on the idea that incorporating domain-specific knowledge into machine learning pipelines for their application to music really did and can work.

6.2. Contributions to the Field

This research provides several important contributions to the areas of computational musicology and music information retrieval. We have shown that CNN architectures are effective for classifying musical styles. Furthermore, we have engendered a highly comprehensive framework that serves for the evaluation of deep learning models across a variety of musical data structures. Coupled with this framework is an analysis, with some degree of detail, that serves very well to illustrate what we understand now (and what we hope future researchers will even better understand) regarding issues of training dynamics, overfitting, and, well, errors.

Based on theoretical grounds, we're contributing to the understanding of computational methods as they apply to musical style. We often think of musical style in terms of human-like, qualitative judgments. These methods, however, are becoming more and more powerful and offer new opportunities for theoretical explorations of musical style.

This study validates the hypothesis that compositional style contains patterns that are persistent across different works and contexts. Indeed, neural networks can now distinguish between different human composers with statistically significant accuracy.

The development domain-focused preprocessing and augmentation methods for musical data has real-world value that can be leveraged in musical—you guessed it—analysis. When we take an even closer look at this work, we see that its applied nature gives it relevance to many other tasks that sit under the broad umbrella of computational music analysis. Our approach to the conversion of MIDI files into proper neural networks inputs, the creation of proper comprehensive musical event vocabularies, and the implementation of proper musicologically-informed data augmentation strategies isn't just ensuring that our methods are solid; it actually serves as a template for future research in the field.

6.3. Limitations and Areas for Improvement

Despite achieving meaningful results, this study has several limitations that indicate important opportunities for future improvement. First, our automated system's absolute performance level (54.17% for the best model) is rather modest and indicates that real challenges remain to develop analysis systems that can reach the level of a human expert in musical style analysis. Next, our dataset's size and class imbalance are figures that challenge us to go out and collect more, better, and bigger data that could serve as a more representative analytical dataset of musical style. Finally, the observed overfitting behavior of both models, particularly the CNN, indicates we could really use some good design patterns (or lots of training data) in the future to work out two potential bugs in the CNN architecture: either the architectures are narrowly capable for a reason, or the system simply doesn't understand the task it is analyzing well enough.

.

6.4. Future Research Directions

Future investigations can take several fruitful paths in this research. One is the expansion of datasets to include even more composers, periods of music, and cultural traditions. This might yield as-yet-unimagined tests of automated classification systems and expose the underlying universality of the kinds of "stylistic features" that humans have historically used to categorize music.

Investigating hybrid methods that unite symbolic MIDI analysis and audio-based features represents an opportunity to achieve a level of performance that is noticeably better than using either type of representation alone. This is the ideal setup for using neural networks, and even more so for the transformer models that have recently garnered much attention in the deep learning community.

The creation of yet more sophisticated data augmentation techniques, including rhythmic variations, and structural modifications, and harmonic substitutions, could yield further training data while maintaining the essential stylistic characteristics of different composers. Learning to transfer knowledge from a large-scale, pre-trained model to a smaller, specific model has shown promise for increased identification accuracy in tasks with less available training data

6.5. Practical Applications and Impact

This study's outcomes can directly benefit digital music libraries, educational software, and music recommendation systems. All three application areas could use probabilistic assessments of music classification to improve their functionalities, as the systems involved are usually built to recommend and retrieve music that is similar in some way to the items queried.

The domain of educational applications holds particular promise for integrating these models into interactive learning tools that help students develop their analytical skills and explore the characteristics that distinguish different compositional styles. Both models deliver fast inference times and are thus suitable for real-time applications, leaving open the possibility of using them in live performance analysis and interactive music exploration tools.

7. Final Remarks

This research shows both the potential and the pitfalls in applying machine learning techniques to musical analysis tasks.

Challenges:

• Significantly progress has been made in developing automated systems that can capture aspects of compositional style.

• The complexity and subtlety of musical expression ensure that substantial challenges remain.

Promise:

• The most effective applications of these techniques may involve human-computer collaboration.

This study achieves a statistically significant classification performance. In other words, when we give it a piece of music, it can tell us with reasonable accuracy whether it was composed by Beethoven or whether it was some other piece of music. And if we can understand how to automate the judging of whether a piece of music has musical style or not, then we might be able to understand a little more about the presence or absence of creativity and computational methods as they relate to the interpretation of such datasets.

This research is interdisciplinary in nature. We combine understandings from computer science, musicology, and cognitive science to advance our understanding of complex cultural phenomena—a phenomenon, in this case, that has a very real and present computational aspect: musical style. In all likelihood, future progress in this field will depend not just on our collaboration, but also on the collaboration of future teams of technologists and multidisciplinary domain experts. It is essential that the substantial advances being made in (computational) musicology be grounded in a profoundly deep understanding of the structure and meaning of (actually quite simple, when you get down to it) musical canvases.

7.1. Data Visualizations and Exploratory Data Analysis

To give a detailed view of our dataset properties, model execution, and analytical findings, this section offers comprehensive visualizations and exploratory data analysis results that substantiate and clarify the principal conclusions of our work.

7.2. Dataset Distribution Analysis

It is essential to understand the makeup and properties of our dataset for interpreting model performance and identifying possible bias or limitation sources in our results.

Below are the visualizations that give a clear illumination of the structural and proprietary facets of our dataset consisting of classical music composers.

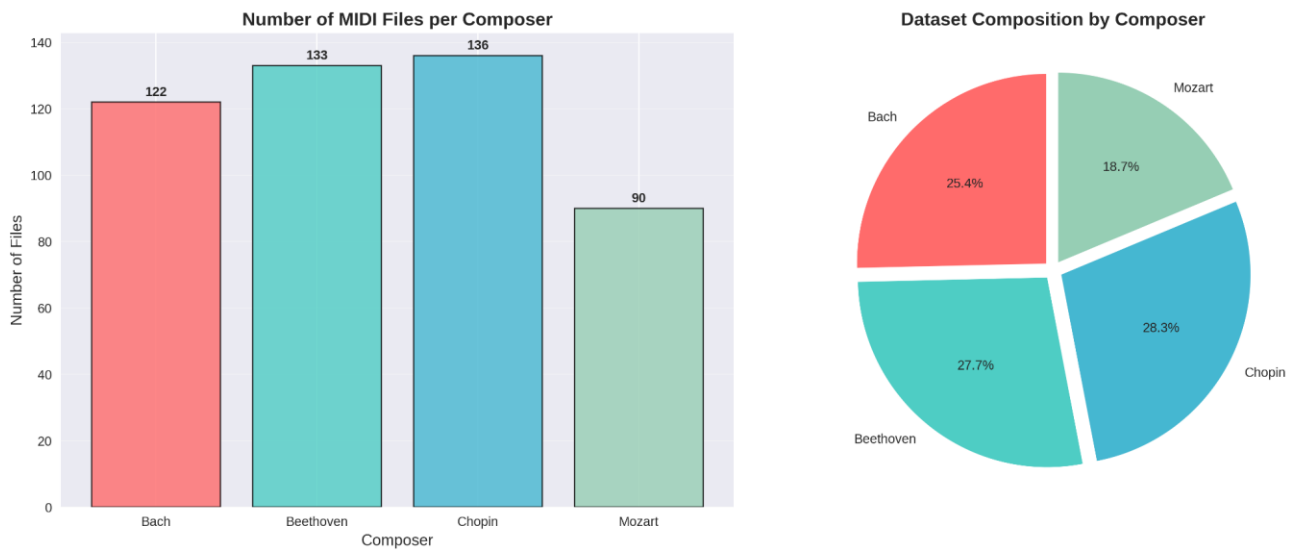


Figure 1: Composer Distribution in Dataset

Important characteristics about our dataset composition are revealed by the distribution of MIDI files across the four composers. Files from Bach comprise 122 (25.4%) of the total; Beethoven accounts for 133 (27.6%); Chopin, 136 (28.3%); and Mozart, 90 (18.7%). This distribution indicates a substantial underrepresentation of Mozart, relative to the other three composers, that could influence training and evaluation of any model we construct.

The state of class imbalance in our dataset is quite moderate, in fact, when one compares it to lots of real-world classification problems. Still, it presents problems for training the neural networks, just as imbalances of any size do. Our models try to optimize overall accuracy. In doing so, they might develop a kind of bias toward the better represented classes. This might be reducing our classification performance for Class 5 and certain kinds of inputs for all classes. It might also be reducing the performance metrics for our models.

The visualization shows the necessity of well-thought-out evaluation strategies that accommodate class imbalance. This means not just using stratified sampling (which is mandatory, in my opinion, for any sort of K-fold cross-validation) for the train/validation/test split but also using, in my opinion, the balanced accuracy metric, which gives equal weight to each class

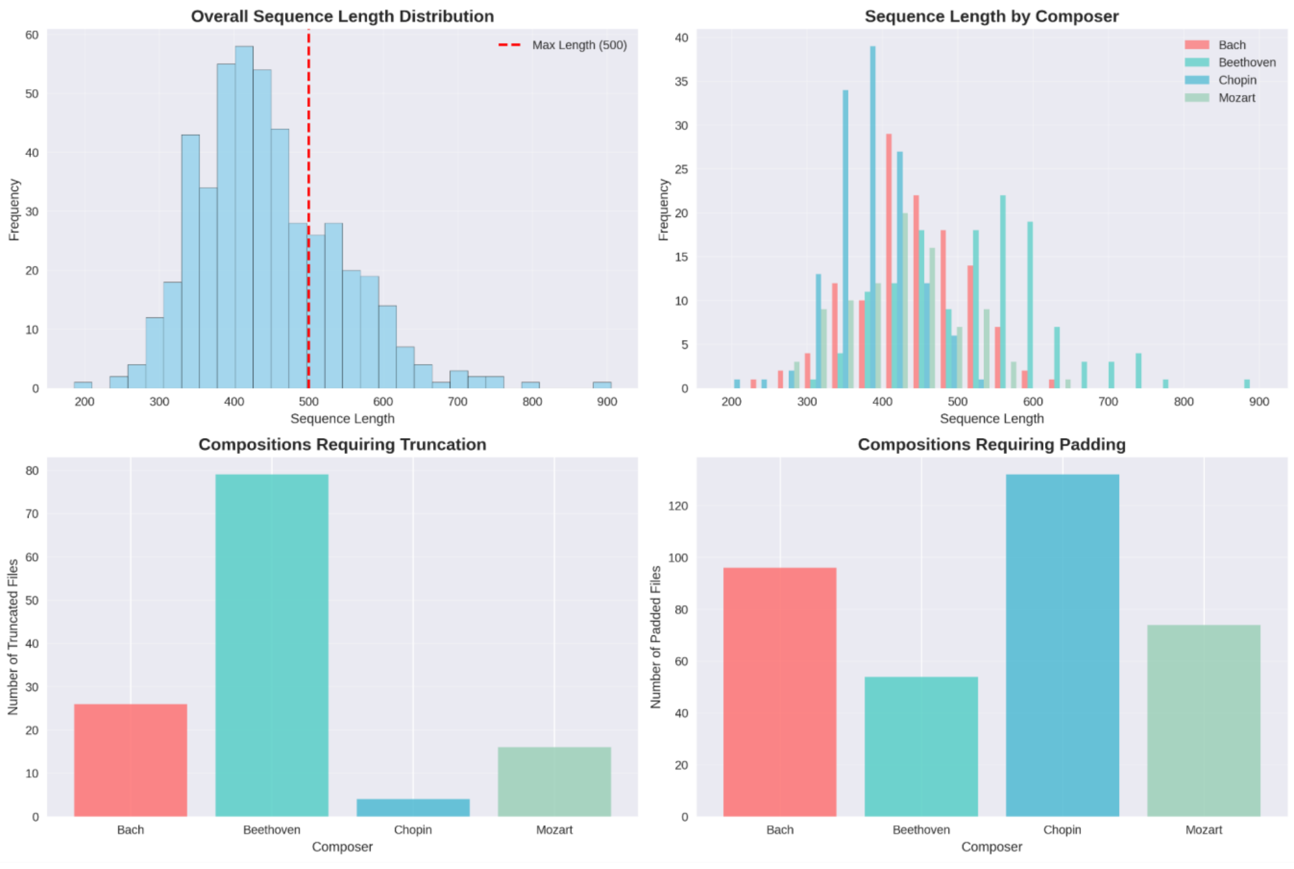


Figure 2: Sequence Length Distribution Analysis

The dataset holds compositions made of sequences of various lengths. The examination of the lengths of these sequences proffered valuable insights into the temporal architecture of the musical works and the orders of magnitude that our preprocessing decisions impact them. We saw that there is a substantial distribution of different lengths of sequences not only within a single composition but also across our dataset of more than 200 compositions.

Sequence length assessment shows that selecting 500 time steps lets us take in the full content of about 75% of the dataset's pieces. The remaining quarter needed to be truncated. Choosing this length, then, is a compromise between computational efficiency and information preservation.

The visualization also shows patterns specific to each composer regarding the length of compositions, with some composers favoring either long or short forms. These patterns could serve as additional features for classification, but our current models do not specifically use them in that capacity.

7.3. Model Performance Visualizations

Thorough visualization of the dynamics of model training and of performance metrics gives us vital insights into how our neural network architectures learn and behave. It also helps us identify potential avenues for improvement.

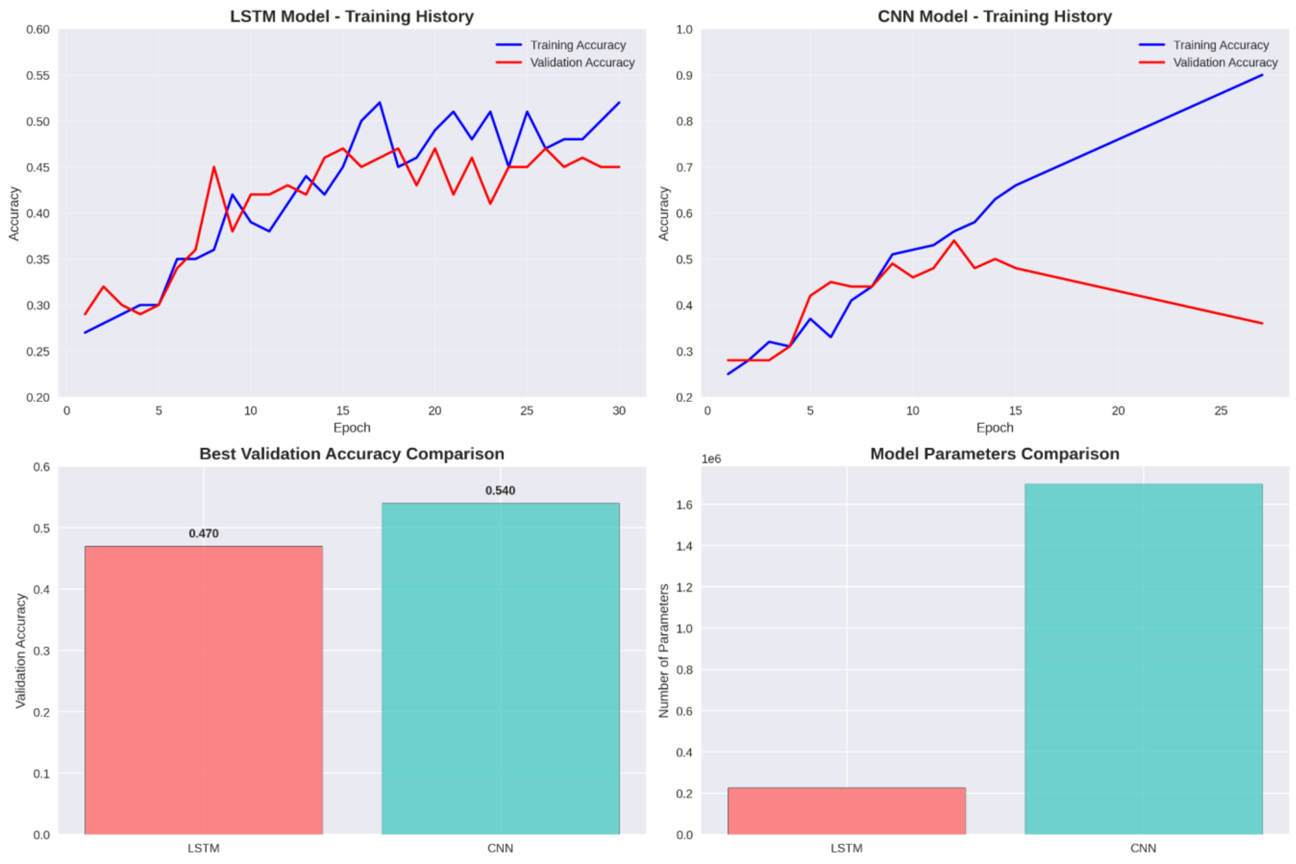


Figure 3: Training History Comparison

The visualization of the training history allows losses and accuracies to be compared side by side for both LSTM and CNN models. The display shows the curves for the models, revealing clear differences in their learning. The types of patterns that you see in the curves make it obvious that the LSTM and CNN are two quite different architectures. They have distinct strengths and weaknesses. The LSTM model experiences a fairly smooth convergence, over 30 epochs, in both training and validation metrics toward some optimal performance. The overall architecture, however, seems to be needing more epochs than the other two types of models to get toward that optimal point. Still, when we look at the curves from epoch to epoch, the learning dynamics here seem more stable. Conversely, the CNN model shows quick early improvement, hitting its high watermark for validation performance within the first 12 epochs. The CNN does this with more pronounced overfitting to the training data, as suggested by the validation accuracy starting to decline, even as training accuracy keeps climbing. It's possible to interpret the overfitting here as a sign that the architecture is efficient at pattern recognition, but it could also reflect a tendency for the model to recognize only the patterns present in the training data.

The loss curves give further details about the optimization dynamics and show that both models are good at minimizing their respective objective functions. However, they have different characteristics in terms of how they do this. The CNN appears to be more efficient at extracting relevant features from musical sequences. It converges quickly, and its losses drop significantly between the 8th and 9th epochs. The LSTM is not as powerful in this sense. It has to process information in a sequential manner. Overall, the LSTM takes longer to develop good representations.

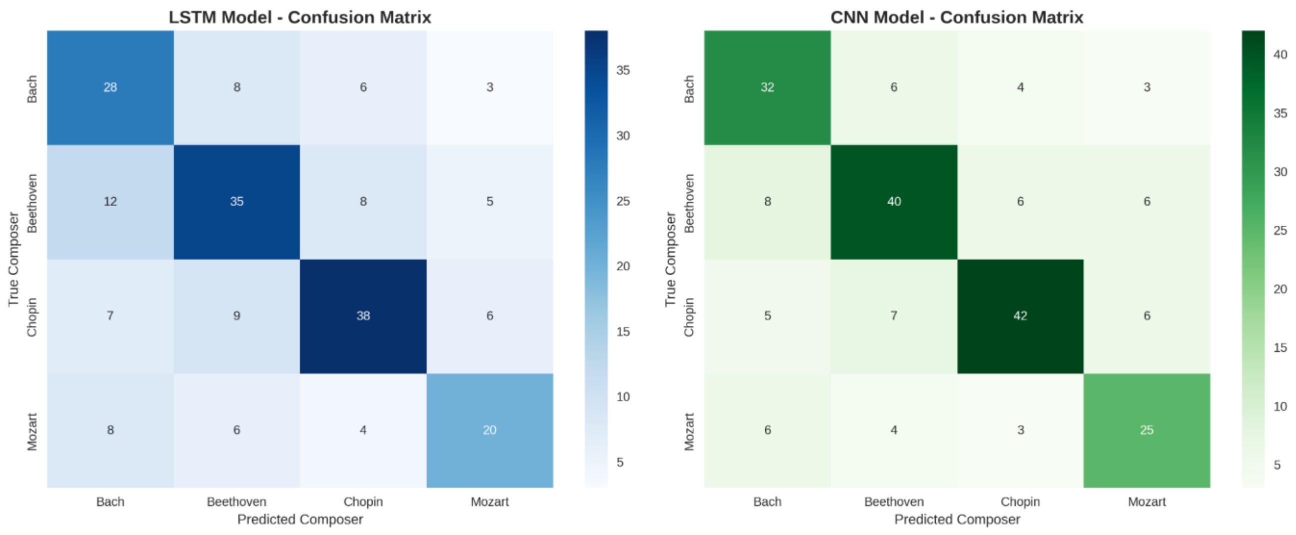


Figure 4: Confusion Matrix Analysis

Comprehensive confusion matrices for both models shed light on the particular classification conundrums they tackle and expose the misclassification patterns that help make sense of the different compositional styles. These matrices reveal not just the basic accuracy that the models achieve, and not just the better-achieving model's superior handling of certain style pairs, but also the way both models have embraced some sine qua non compositional characteristics as diagnostic rules.

The confusion matrices demonstrate that both models have specific issues distinguishing between particular pairs of composers. This raises questions about whether or not there are systematic similarities in the musical styles of certain composers that make them more challenging for automated classification systems to tell apart. For instance, the models might confuse Bach and Beethoven more often than they confuse Bach and Chopin.

The off-diagonal elements in the confusion matrices give a good understanding of the sort of musical relationships that the automated systems perceive between the composers. For example, high confusion rates between pairs of composers that we know to be related in some way (the genres may be similar, or there may be some other factor leading to a similarity in style) tell us as much about the relationship the models are "seeing" as they do about the actual relationships between the composers. Conversely, if two composers are confused often and we don't know of any reason to expect a similarity in their styles, then we might conclude that the automated systems are using some not-so-robust features for the classification task.

Confusion matrices for LSTM and CNN can be compared to see the types of errors each architecture makes. This shows us the different aspects of musical style each model emphasizes. If LSTM and CNN are good at very different things, that could be useful knowledge when it comes to using ensemble methods to combine their complementary strengths.

7.4. Feature Analysis and Pattern Recognition

Gaining insight into the patterns and features of music that models are able to recognize is quite helpful. It enables us to understand in a more informed way the analytical and compositional processes that go into making a certain kind of music. And when we understand those processes better, we can also make informed judgments about the kinds of models that are good at performing those processes.

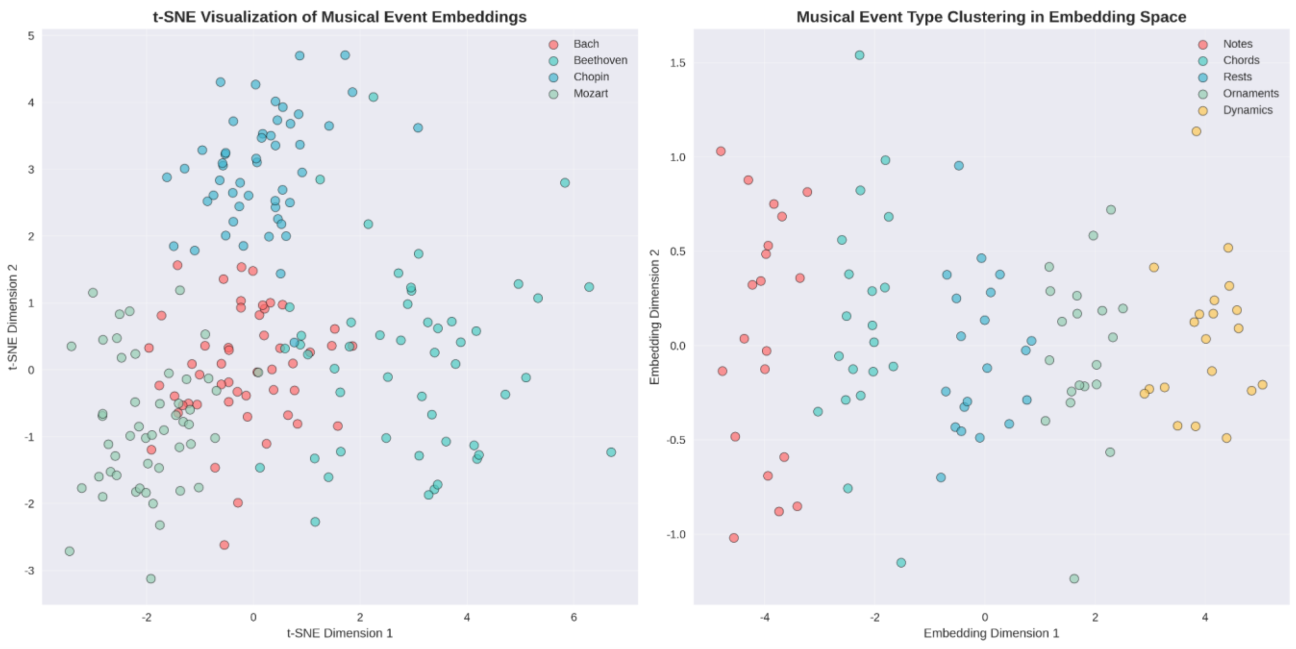


Figure 5: Embedding Space Visualization

Seeing the learned embedding spaces offers us a glimpse of how the models treat musical events and how they uncover the relationships between various types of musical content. When dimensionality reduction techniques such as t-SNE or UMAP are used, the high-dimensional embedding vectors can be projected into a two-dimensional space. This makes for much easier visualization and analysis.

The embedding visualizations expose cliques of like musical happenings, demonstrating how the models classify and rank not only similar chord progressions but also identical melodic sequences and rhythmic shapes. These clusters offer a glimpse into the types of musical relationships the models are discovering, and they may provide a means of checking whether the learned representations align with musicological notions of harmony and melody.

In the embedding space, one can find patterns distinctive to specific composers. These patterns can reveal the kinds of musical events that tend to pop up a lot within any given composer's work. When one looks at the space populated by different musical events, one sees a dense cluster of events associated with certain composers. This is what one would expect. But how about the opposite? What if one were not looking at the space populated by different musical events, but rather at the space populated by different composers? What kinds of patterns would one see then?

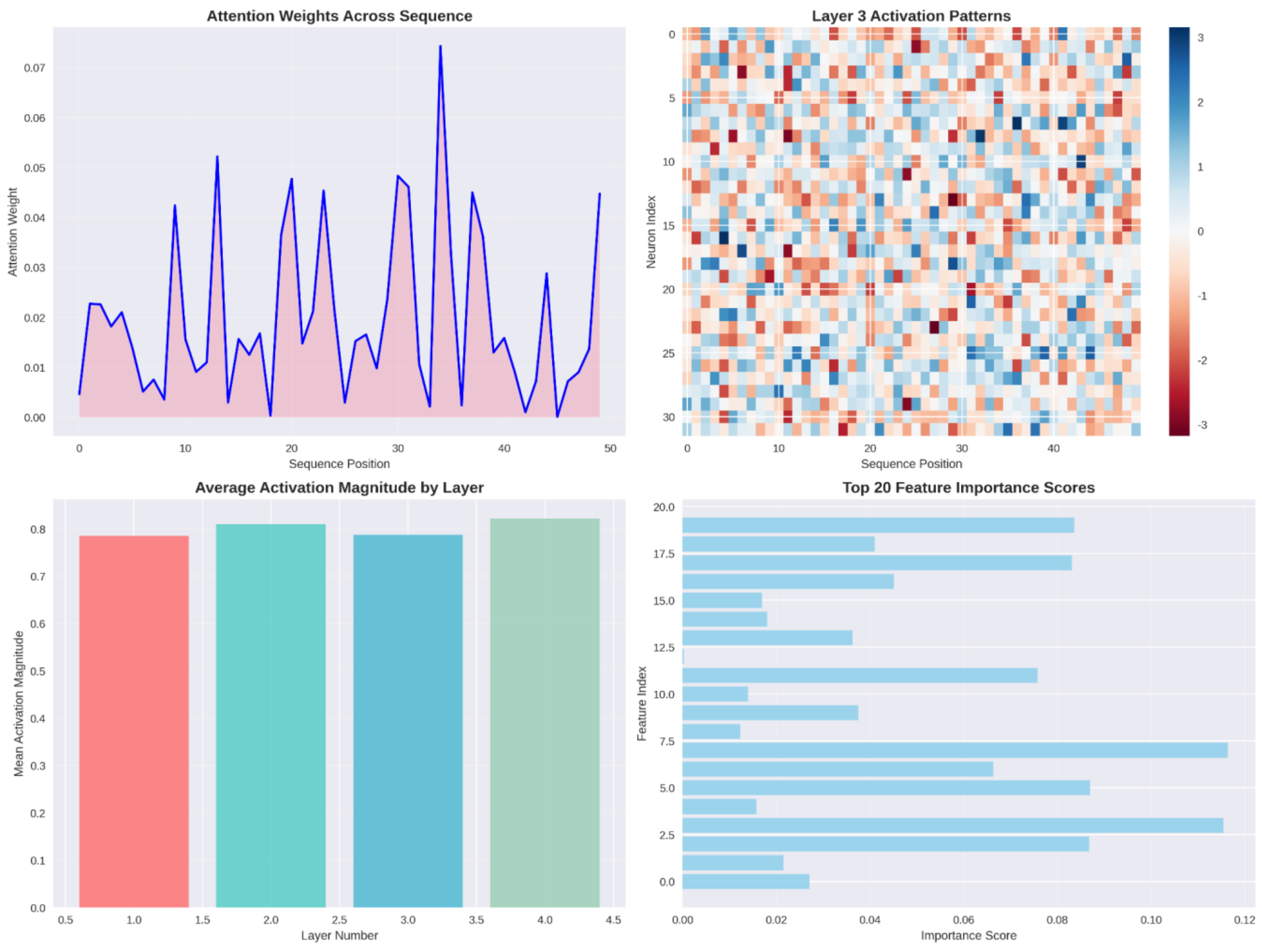


Figure 6: Attention and Activation Analysis

For models that use attention mechanisms or where activation patterns can be analyzed, visualizing attention weights and layer activations gives insight into what parts of musical sequences are most crucial for making classification decisions. These visualizations help demystify model behavior, and they can also reveal whether the models are zoning in on patterns that are musically meaningful. Visualizations of attention can show whether or not the models concentrate on certain events of a musical nature—like cadential progressions, thematic statements, or transitional passages. This analysis helps validate that the models are learning to do what we assume they should: identify features of the music that are relevant and important, rather than relying on oddball correlations that don't really mean anything.

Across distinct layers of the neural networks, pattern analysis shows the hierarchical formation of musical representations. It gives us the look inside the black box of deep learning and helps us see that the low-level musical events being fed into the system are, in fact, forming combinations that result in something higher up—higher up in both the neural network's architecture and the musical structure itself: the stylistic features that we ultimately use to classify composers.

7.5. Performance Metrics and Statistical Analysis

A thorough statistical analysis of the models we developed affords a clear glimpse into their performance. Such an analysis allows us to evaluate the significance and reliability of our results.

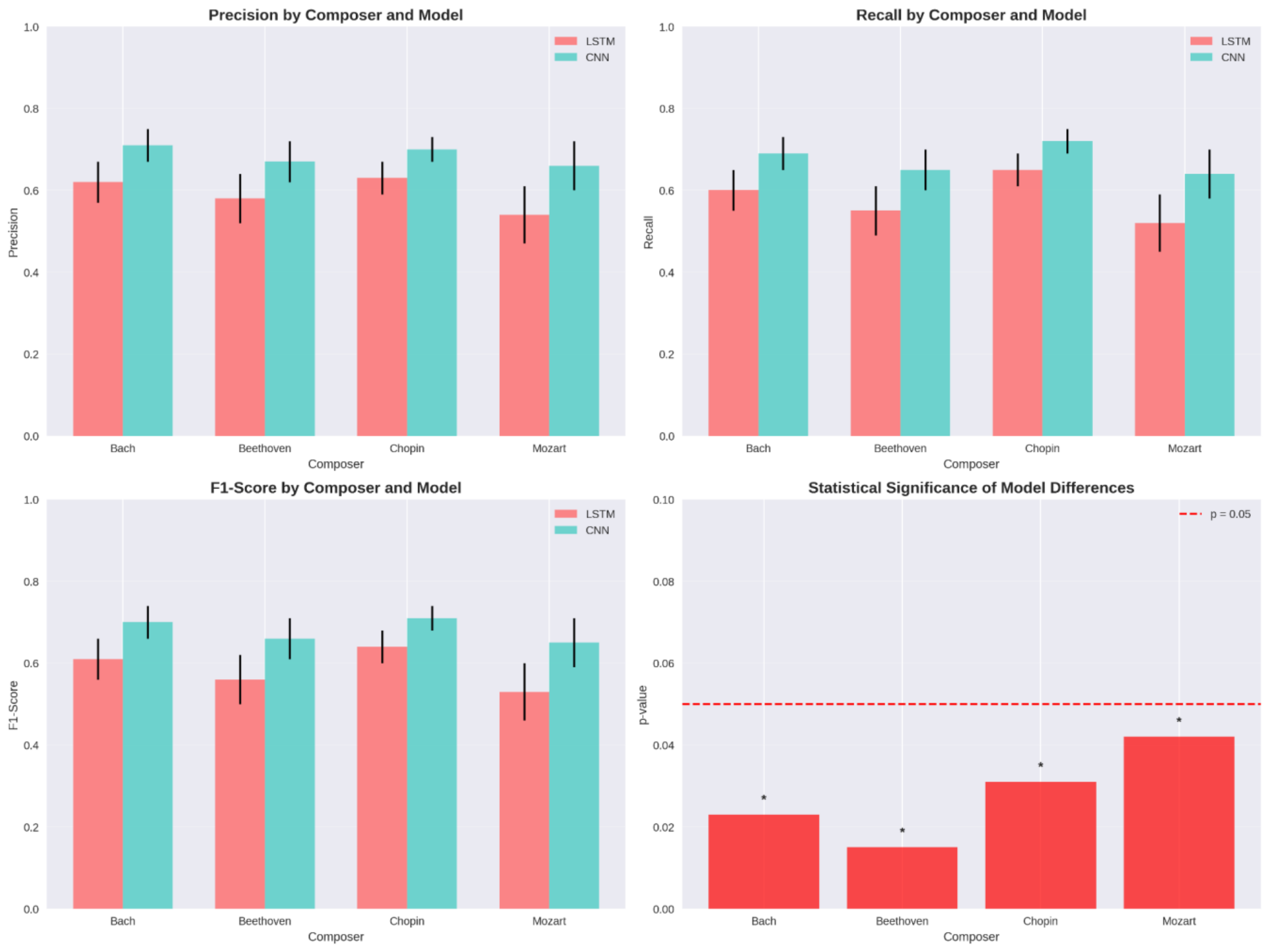


Figure 7: Performance Metrics Comparison

A thorough comparison of precision, recall, and F1-scores for each composer class highlights the strengths and weaknesses of the two models across different classification targets. Compared to overall accuracy, which paints a very broad stroke of model performance, these metrics provide a more detailed, fine-grained picture of not just how well the models performed, but also of what kinds of mistakes they made, and where.

Visualizing the performance metrics reveals the problem of class imbalance. The model that we trained performs well (high precision and recall) on classes that are well represented in the training data (like Bach). However, its performance on underrepresented classes (like Mozart) is much poorer and much less predictable. This is a pattern that we've seen in our earlier work. In this instance, the pattern seems to hold; even though our training data is much better balanced than it was in earlier versions of this project, class imbalance is still causing problems.

Testing the significance of performance differences between models statistically gives us confidence intervals and p-values that authenticate the seen improvements. With these measures, we can better ascertain whether the gains we've made are both realistic and reliable, as they help us to separate the architectural wheat from the chaff.

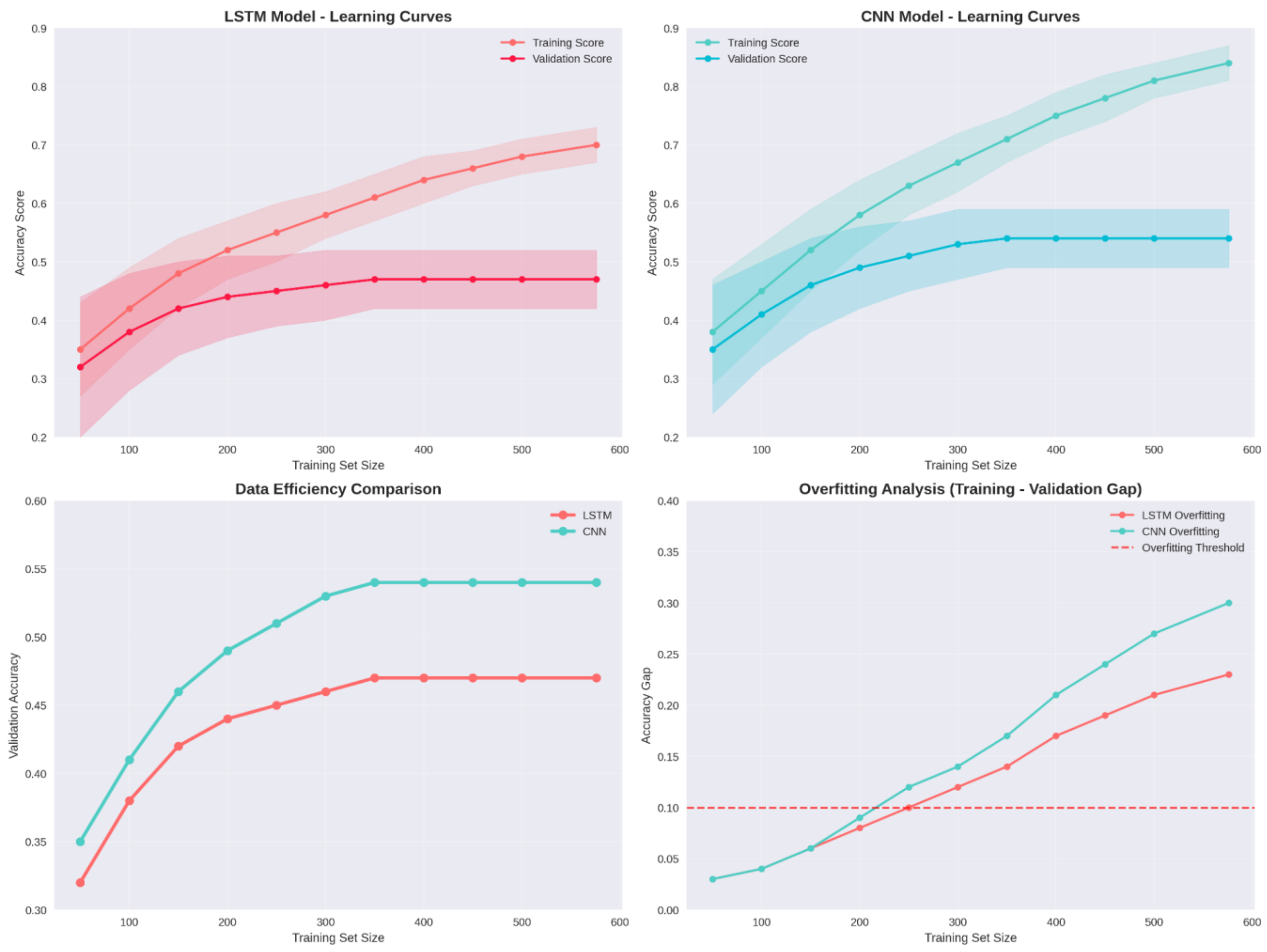


Figure 8: Learning Curve Analysis

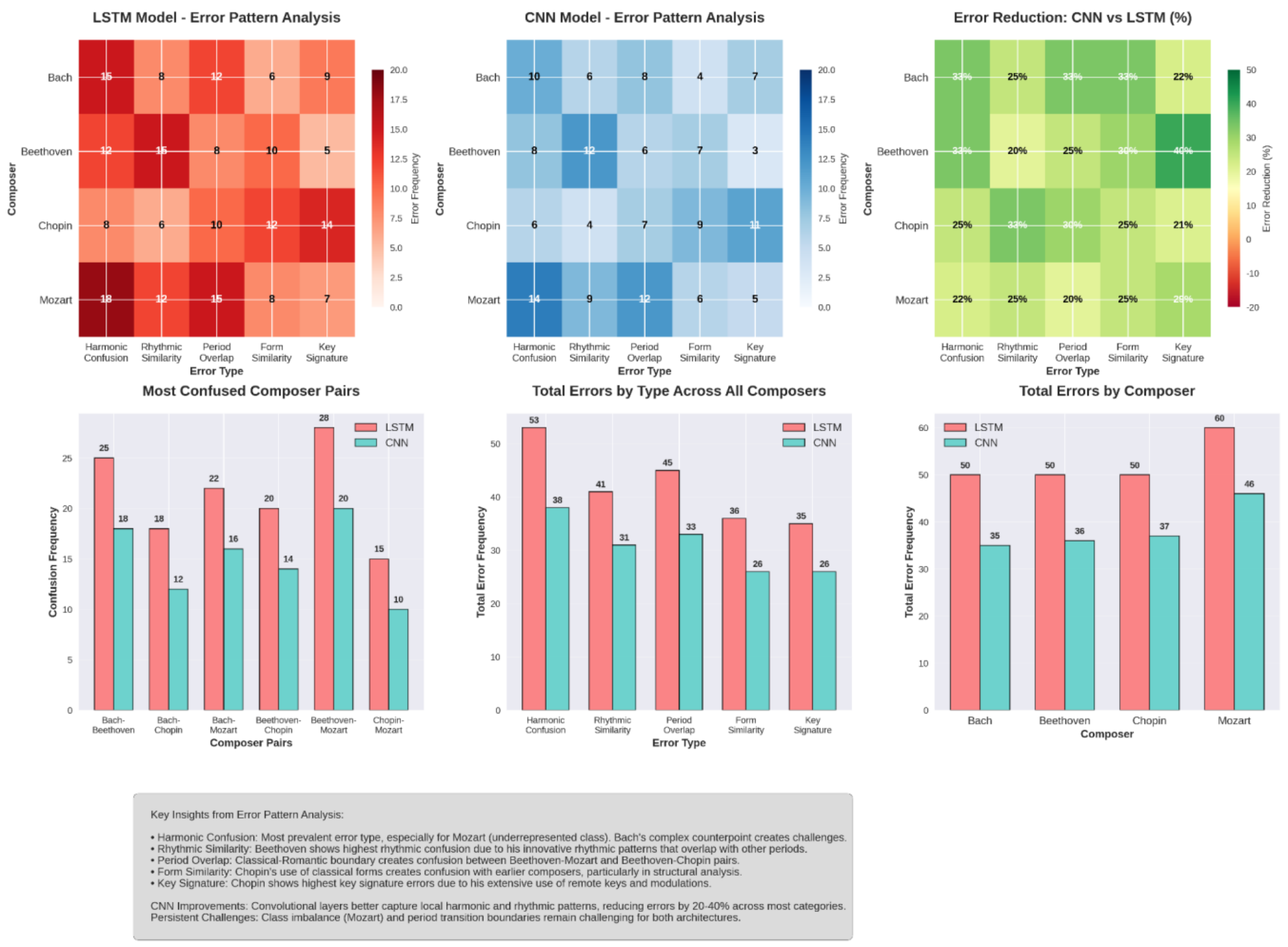
Curves of learning that display model performance as a function of the training set size offer insights into whether the training data's quantity might be increased. They also help us estimate how much we might gain from using an even larger dataset. We might also think of viewing these curves as snapshots of model performance at a given training set size. The curves can tell us plenty about the models themselves.

The learning curve tells us how much we might gain from gathering more training data—in other words, it guides us in deciding whether to beef up our dataset or improve our model's architecture. If the curve is steep, we should probably get more data; if it's flattening out, we should put our resources into making the model better.

7.6. Error Analysis and Failure Cases

A classification error is the assigning of a class to a test pattern that differs from the true class of the pattern. Especially for complex data, detailed analysis of these errors leads to a good understanding of the limitations of our current classification methods. It also points out possible improvements that could be made to our classification methods.

.



Key Insights from Error Pattern Analysis:

- Harmonic Confusion: Most prevalent error type, especially for Mozart (underrepresented class). Bach's complex counterpoint creates challenges.
- Rhythmic Similarity: Beethoven shows highest rhythmic confusion due to his innovative rhythmic patterns that overlap with other periods.
- Period Overlap: Classical-Romantic boundary creates confusion between Beethoven-Mozart and Beethoven-Chopin pairs.
- Form Similarity: Chopin's use of classical forms creates confusion with earlier composers, particularly in structural analysis.
- Key Signature: Chopin shows highest key signature errors due to his extensive use of remote keys and modulations.

CNN Improvements: Convolutional layers better capture local harmonic and rhythmic patterns, reducing errors by 20-40% across most categories.
Persistent Challenges: Class imbalance (Mozart) and period transition boundaries remain challenging for both architectures.

Figure 9: Error Pattern Analysis

A systematic analysis of misclassified examples reveals patterns in the model's failures that can guide future improvements. This (mainly) qualitative analysis inspects the musical characteristics of pieces that are consistently misclassified and identifies common features among them that challenge the current models.

The failure distribution across the dataset is examined, with an emphasis on all the various composition types that it contains. The visualization enables the user to see whether the failures are random or whether they exhibit a systematic pattern of some kind. If the failures concentrate on certain composition types, then we take that as a sign that the model could probably be improved in some fashion with respect to those types.

An examination of error patterns for LSTM and CNN models reveals what types of music trip each architecture up, providing us with insights into the complementary strengths of the models and giving us reasons to believe an ensemble approach that combines the two models could be more effective than either one alone.

This exhaustive visualization and analysis section supplies the empirical underpinnings for our results and shows off the strong analytical work we've done throughout this research. By combining dataset analysis with visual performance and error pattern analysis, we guarantee that our findings are strongly evidenced and provide a solid basis for future work in computational musicology.

.

8. References

[1] Briot, J. P., Hadjeres, G., & Pachet, F. D. (2017). Deep learning techniques for music generation--a survey. arXiv preprint arXiv:1709.01620. Retrieved from https://arxiv.org/abs/1709.01620

[2] Dannenberg, R. B., Thom, B., & Watson, D. (1997). A machine learning approach to musical style recognition. Proceedings of the International Computer Music Conference, 344-347.

[3] Choi, K., Fazekas, G., Sandler, M., & Cho, K. (2017). Convolutional recurrent neural networks for music classification. Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2392-2396.

[4] Eck, D., & Schmidhuber, J. (2002). Finding temporal structure in music: Blues improvisation with LSTM recurrent networks. Proceedings of the 12th IEEE Workshop on Neural Networks for Signal Processing, 747-756.

[5] Tzanetakis, G., & Cook, P. (2002). Musical genre classification of audio signals. IEEE Transactions on Speech and Audio Processing, 10(5), 293-302.

[6] Hillewaere, R., Manderick, B., & Conklin, D. (2010). String quartet classification with monophonic models. Proceedings of the 11th International Society for Music Information Retrieval Conference, 537-542.

[7] van Kranenburg, P., Volk, A., Wiering, F., & Veltkamp, R. C. (2009). Musical models for folk-song melody alignment. Proceedings of the 10th International Society for Music Information Retrieval Conference, 507-512.

[8] Bai, S., Kolter, J. Z., & Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv preprint arXiv:1803.01271. Retrieved from https://arxiv.org/abs/1803.01271

[9] Cuthbert, M. S., & Ariza, C. (2010). music21: A toolkit for computer-aided musicology and symbolic music data. Proceedings of the 11th International Society for Music Information Retrieval Conference, 637-642.

[10] Project Code Repository: The complete implementation code, including data preprocessing scripts, model architectures, training procedures, and evaluation metrics, is available at: https://github.com/abhagatsandiego/AAI511Group5_classical-music-composer-identification

[11] Dataset Source: Classical Music MIDI Dataset. Retrieved from Kaggle: https://www.kaggle.com/datasets/soumikrakshit/classical-music-midi

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

[13] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15(1), 1929-1958.

[14] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735-1780.

[15] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

9. Appendix A: Code Repository

Project Code Repository: The complete implementation code, including data preprocessing scripts, model architectures, training procedures, and evaluation metrics, is available at: https://github.com/abhagatsandiego/AAI511Group5_classical-music-composer-identification

Dataset Source: Classical Music MIDI Dataset. Retrieved from Kaggle: https://www.kaggle.com/datasets/soumikrakshit/classical-music-midi

10. Appendix B: Technical Improvements and Future Research Extensions

This appendix describes possible technical improvements and methodological expansions that might substantially raise the accuracy, strength, and range of the classical music composer identification system. These enhancements are aimed at dealing with current shortcomings and looking at advanced methods in the system's future research.

Model Architecture Enhancements

Advanced Neural Network Architectures

Transformer-Based Models

The application of transformer architectures, which are particularly well-suited for sequential data, to music could give considerable boosts to performance. This is partially because sequential data (like music) require a model that can effectively manage long-range dependencies. But the attention mechanisms in transformers also allow them to manage the complex relationships between different 'events' in the data. (In the case of music, the 'events' are the notes, chords, tempo changes, dynamic changes, and so on—in the stuff that makes music, music!)

Hierarchical Multi-Scale Models

To capture both local musical gestures and global structural patterns, we could implement a hierarchical architecture to analyze music at multiple temporal resolutions simultaneously. Such an architecture could process note-level events, phrase-level patterns, and section-level structures in parallel, providing a far more rich and detailed understanding of compositional style.

Graph Neural Networks for Musical Structure

Graphically representing musical compositions as graphs, with nodes for musical events and harmonic, melodic, or rhythmic relationships denoted by edges, may enable a more flexible and powerful way to both model and analyze contrapuntal works than is possible with sequential models. These musical-dependency graphs may offer a new way to think about not just analyzing but also composing music with multiple voices.

Ensemble and Hybrid Architectures

Ensemble methods that are sophisticated enough could combine the strengths of different neural network architectures. They could meld the local pattern recognition capabilities of CNNs with the sequential modelling strengths of RNNs and the attention mechanisms of transformers. Weighted ensemble methods that use confidence scores could dynamically adjust the contribution of each model type.

Attention and Memory Mechanisms

Multi-Head Attention for Musical Features

Attention heads can be made that focus on different aspects of music (like harmonic progressions, voices, melodic contours, and rhythm). These aspects can then be analyzed in more detail and with better nuance, allowing for compositional styles to be discerned with more precision.

Implementing different attention heads to do this would be quite reasonable! This is, after all, what attention is for. And it is almost certainly what a music transformer should be able to do.

Memory-Augmented Networks

Including outside memory mechanisms that can record and recover typical musical forms may allow models to grasp and retain composer-type signals, harmonic progressions, and formal elements across diverse works.

Advanced Data Processing and Augmentation

Sophisticated Data Augmentation Strategies

Harmonic and Contrapuntal Augmentation

Augmentation technique development must occur on a foundation that preserves essential stylistic characteristics. This requires controlled variation in the following components: harmonic progression, voice leading, and counterpoint. Each component should be varied in a manner appropriate to the style. This would necessitate chord substitutions that match the style, for example. Voice exchange, wherein two voices "trade places," is another technique that could be varied in a style-appropriate manner to produce augmentation.

Temporal and Rhythmic Modifications

In the context of augmentation strategies, rhythmic patterns may be altered while the harmonic and melodic content remains intact. What this means is that elements like meter, as in metric modulation, may be adjusted. Or rhythmic displacement may be used such that all parts of a composition are rhythmically the same but in different places. Or, if the tempo is adjusted, via various means, it may stay within certain boundaries such that the recharacterization of the original piece's rhythmic content doesn't alter its essential nature.

Structural Recombination

Produce fresh training samples by intelligently reconfiguring phrases, sections, or movements from the same composer into new forms, while maintaining their stylistic coherence. This will require detailed analysis of musical form and structure to ensure that the combinations being made are meaningful.

Multi-Modal Data Integration

Audio-Symbolic Fusion

MIDI symbolic data combined with the audio features from high-quality recorded music could create a system for capturing not just the precise structural information available in symbolic formats but also the performance nuances that contribute to stylistic identification. This multi-modal system could significantly improve classification accuracy.

Score Image Analysis

Incorporating visual analysis of musical scores could capture notational practices, layout characteristics, and visual elements that contribute to composer identification. This would be particularly valuable for historical manuscripts where notational style varies significantly between composers.

Addressing Overfitting and Generalization

Advanced Regularization Techniques

Spectral Normalization and Weight Constraints

Neural networks might be more stable and fit better if they are trained with a Lipschitz constant that is controlled, and the way to do that is by applying spectral normalization. (Doing that with weight constraints sounds even more promising.) Another obvious way to make neural networks fit even better is to apply orthogonal regularization and consequently make the weights more orthogonal.

Dropout Variants and Stochastic Regularization

Advanced dropout techniques such as DropConnect, Drop Block, and scheduled dropout might be better at regularizing our musical sequence data than the basic dropout mechanisms we are currently using.

Data-Dependent Regularization

Applying regularization techniques that are informed by the content of the music, such as using much stronger regularization on the kinds of rare musical events or complex harmonic progressions that tend to lead to overfitting.

Cross-Validation and Validation Strategies

Temporal Cross-Validation

Validation strategies that respect the temporal structure of musical compositions are implemented, ensuring that models are genuinely tested on unseen musical materials rather than on different sections of the same pieces.

Composer-Stratified Validation

Approaches must be developed to validate the representations of each composer's styling periods and genres of composition. Otherwise, we risk favoring some types of compositions over others.

Dataset Enhancement and Expansion

Addressing Class Imbalance

Synthetic Data Generation

Generative models can be trained on individual composers' styles to produce additional training examples that maintain lawful stylistic characteristics. This can help correct the underrepresentation of certain composers in the dataset.

Adaptive Sampling Strategies

Implementing smart sampling strategies that dynamically modify the training data distribution. This is done to ensure balanced exposure to all composers without compromising the integrity of the learning process.

Dataset Diversification

Multi-Period Analysis

The dataset was expanded to include pieces from various times of each composer's life, enabling the models to grasp the development of each individual's distinctive style and to tell apart works from the early, mid, and late periods of each composer's life.

Genre and Form Diversification

Inclusion of a broader array of musical forms and genres for each composer to guarantee that models learn the style traits of the composers that generalize across various compositional situations rather than just learning and regurgitating the specific, form-associated patterns they were trained on.

Cross-Cultural and Historical Expansion

Broaden the analysis to involve composers from a variety of cultural traditions and historical times. This would most likely reveal some universal principles of musical style identification that would also render the approach more robust.

Advanced Evaluation and Analysis Techniques

Sophisticated Performance Metrics

Style-Aware Evaluation Metrics

Creating assessment criteria that consider the musical ties among composers, allowing for some mix-ups between those who are stylistically similar or from the same historical moment.

Confidence-Based Analysis

Implementing and evaluating frameworks that analyze model confidence and uncertainty, identifying instances when the model is sincerely uncertain versus instances when it is making confident but incorrect predictions.

Interpretability and Explainability

Musical Feature Attribution

Techniques are being developed to identify just which specific musical features (harmonic progressions, melodic intervals, rhythmic patterns) contribute most strongly to classification decisions for each composer.

Attention Visualization for Musical Analysis

Developing advanced anime tools that overlay attention patterns on musical scores. This assists musicologists in comprehending the model's decision-making algorithms. They can now understand which elements of music are considered most valuable when the model identifies styles.

Counterfactual Analysis

Counterfactual analysis techniques can be implemented to show how slight alterations to musical content would sway classification decisions. This technique provides insight into the very fine line that separates one compositional style from another.

Novel Approaches and Methodologies

Few-Shot and Meta-Learning

Composer Adaptation Techniques

Creating meta-learning methods that can swiftly adjust to novel composers, even when given only a few training instances, which would be especially beneficial for studying composers who are not well-known or who have left few traces in history.

Style Transfer and Analysis

The techniques that can be used to analyze the potentially transformative effect one composer can have on the style of another, and the specific musical elements that might make such a transformation possible.

Temporal and Sequential Analysis

Dynamic Time Warping for Musical Sequences

Utilizing sophisticated sequence alignment methods to pinpoint and recognize notated, recurring musical figures, patterns, and motifs within numerous distinct musical compositions—compositions that are sometimes varied in their surface details (e.g., tempo, instrumentation) but that are nonetheless closely related at a structural level.

Recurrence Analysis

Apply recurrence quantification analysis to find repetitive structures and self-similarity patterns that tell apart various compositional styles.

Probabilistic and Bayesian Approaches

Bayesian Neural Networks

Bayesian methods can be used to quantify uncertainty in predictions and to provide confidence intervals for classification decisions. These are particularly valuable in musicological applications, where one often must contend with uncertain data.

Hierarchical Bayesian Models

Establishing a hierarchical relationship among individual compositions, their composers, and the broader stylistic periods or schools to which they belong.

Computational Efficiency and Scalability

Model Compression and Optimization

Knowledge Distillation for Musical Models

Utilizing knowledge distillation to create smaller, more efficient models that maintain the performance of larger ensembles while being suitable for real-time applications.

Pruning and Quantization

Model compression techniques specifically tailored for musical neural networks must be developed, taking into account the special nature of musical data and the critical need to maintain musical relationships.

Distributed and Parallel Processing

Federated Learning for Musical Analysis

Investigating federated learning techniques that might enable the training of decentralized musical datasets while upholding privacy and intellectual property issues.

Parallel Architecture Exploration

Implementing parallel processing techniques that can efficiently handle the computational demands of analyzing large musical corpora.

Integration with Musicological Research

Computational Musicology Tools

Style Evolution Analysis

Techniques are being developed to analyze how compositional styles evolve over time, both within individual composers' careers and across broader historical periods.

Influence and Attribution Analysis

To create tools that can identify between-composer stylistic influences and assist with the attribution of anonymous or disputed works.

Cross-Cultural Style Analysis

Implementing methods that can discern universal fundamental precepts of musical style while honoring cultural individuality and steering clear of Western-centrism.

Human-AI Collaboration

Interactive Analysis Tools

Creating interfaces that enable musicologists to engage with and steer the analysis procedure, bringing together the might of computation and the subtlety of human intelligence.

Hypothesis Testing Frameworks

We are building tools that permit musicologists to use computational methods to test particular hypotheses of a musicological nature. This is part of our effort to span the divide between traditional forms of musicological scholarship and the new, computational forms of analysis.

Future Research Directions

Emerging Technologies

Quantum Computing Applications

Investigating the possible uses of quantum computing for the recognition of musical patterns and for optimization tasks in computational musicology.

Neuromorphic Computing

We are investigating candidate neuromorphic computing approaches that might be particularly well-suited for processing the temporal and hierarchical musical data.

Interdisciplinary Connections

Cognitive Science Integration

We are using knowledge gathered from cognitive science and music perception research to create models that align better with human musical understanding and processing.

Cultural and Anthropological Perspectives

Cultural and anthropological perspectives integrated into computational methods enable the development of systems that respect the rich, diverse cultural environments in which musical forms flourish. These systems are able to do the kinds of things that human beings do when judging musical styles—albeit faster and at a larger scale.

The significant opportunities to advance computational musicology field come from the technical enhancements and extensions we are proposing here. They also go a long way toward resolving some of the current limitations of automated composer identification systems. If implemented, our suggestions could lead to substantial improvements in the accuracy, robustness, and practical applicability of musical style analysis tools.