Research Article

Classifying Classical Music Genres with Neural Networks

Maggie Liu

BASIS Independent Silicon Valley, 1290 Parkmoor Avenue, San Jose, CA, USA

ABSTRACT

Current neural network models can process and interpret music for tasks such as melody completion and genre or style classification. However, previous classification tasks do not account specifically for distinct composition styles of different classical music periods, often focusing instead on modern genres. To bridge this gap, this project investigates the use of natural language processing techniques to classify musical excerpts from the Baroque, Classical, and Romantic periods. A curated dataset of samples representative of the three eras, converted to the OctupleMIDI format, was used to train a Sentence Transformers model to complete the classification task with maximum accuracy—62.5% when trained on all three categories and 90.5% when the Classical and Romantic labels were merged. These results indicate that the model was most effective at distinguishing Baroque music, suggesting clearer stylistic separation. These findings demonstrate the feasibility of using sentence-level embeddings for symbolic music classification, offering potential applications in musicological analysis, genre tagging for recommendation systems, and quantitative exploration of musical style beyond human perception.

Keywords: Machine Learning; Classical Music; Subgenre Recognition; Genre Classification; Music Information Retrieval; Natural Language Processing

INTRODUCTION

Different forms of neural networks, such as recurrent neural networks (RNNs) and convolutional neural networks (CNNs), have long been used to accomplish tasks from sentiment classification of a sentence to detecting objects in an image (1). Neural networks can interpret human language by identifying patterns in large amounts of data, forming natural language processing (NLP) methods. The large body of unlabeled music data available can be translated into embeddings, or numerical representations, in a fashion similar to how unlabeled natural language is processed in NLP methods.

Building upon this foundation, within the domain of music, a variety of tasks including melody completion, harmony generation, and genre and style classification have been accomplished (2).

However, these tasks often center around modern genres like pop and rock, while applications within classical music have been limited. Identifying classical music periods can be a difficult and inconsistent task for humans; a study compared music history students with

Corresponding author: Maggie Liu, E-mail: maggie08.liu@gmail.com.

Received April 21, 2025**; Accepted** May 17, 2025
https://doi.org/10.70251/HYJR2348.335159

untrained individuals in determining the period a piece comes from. The results demonstrated that both groups rely on factors like rhythm and on the ordering of the excerpts to determine the relative chronology of different compositions (3). Moreover, the divisions between different styles cannot always be clearly defined by year; the lines between periods are often blurred in the overlaps between chronological eras (4). Thus, specific composers or pieces in these junctions can only be classified based on their traits rather than strictly by year of composition.

As a result, this project investigates the use of natural language processing models to classify symbolic music excerpts into three key classical music eras—Baroque, Classical, and Romantic. A dataset of MIDI files was compiled, labeled by period, and converted into the OctupleMIDI format to represent relevant musical features numerically. A Sentence Transformers model was then trained on this data, adjusting factors such as learning rate and training set size to attain maximum classification accuracy. By leveraging the pattern recognition capabilities of machine learning, this approach offers potential applications in helping music historians distinguish overlapping compositional styles, as well as in enhancing sub-genre tagging within music catalogues and recommendation systems. Ultimately, this classification task aims to bridge the gap between computational methods and classical music analysis by investigating stylistic differences through quantitative embeddings rather than subjective human-identified trends.

LITERATURE REVIEW

Background research began with an overview of the Transformer architecture, introduced in the 2017 paper "Attention Is All You Need," that enables the attention mechanism that weights elements in an input by relevance, much like how humans focus on certain parts of a sentence while reading (5). This approach is critical in the domain of music because each note in an excerpt is connected to the context of the values around it, an idea that models like PopMAG inherently utilize through tracking interdependencies within a piece of music (6). These links are crucial for the formation of a sensible piece of music. As it pertains to genre classification within classical music, the context surrounding each smaller section of an excerpt is crucial in revealing trends throughout the piece, contextualizing it within the musical era to which it belongs through common patterns in pieces of the same period.

One neural network designed for processing music is MusicBERT, based upon the BERT model, or Bidirectional Encoder Representations from Transformers, developed in 2018 by Google AI Language for NLP tasks (7). Through testing several Google CoLab notebooks, one can clearly access the features BERT has to offer, including sentence completion (given a masked word in a sentence, BERT can come up with several possible or probable completions) and sentiment classification (BERT can tag the tone of a statement as positive or negative) (8, 9). Trained on a large amount of data, BERT takes advantage of the Masked Language Model and Next Sentence Prediction to infer a missing word or a logical next step respectively; additionally, it utilizes the aforementioned Transformer architecture (7).

MusicBERT has strong genre classification capabilities, outperforming previous models like melody2vec and the more recent PiRhDy on classification tasks (2). However, it is not explicitly designed to interpret classical music. Similarly, PopMAG is specified to only deal with popular music genres (6).

Within the domain of classical music, the Chamber Ensemble Generator has successfully expanded on the limited data available by generating new realistic excerpts of four-part Bach chorales; however, its limitation lies in that it is only capable of dealing with Bach chorales and no other styles of classical music (10).

One use case of MusicBERT is the Midiformers project, which enables parts of music tracks to be predicted after being masked, similar to the sentence classification task given to the standard BERT model (11). The project notebook illustrates the steps to use MusicBERT, including conversion from standard MIDI format to OctupleMIDI and loading the BERT and MusicBERT models. It also includes demonstrations of the masking task (12).

Previous research applying neural networks generally represents music in a MIDI (Musical Instrument Digital Interface) format. MIDI representation permits a smaller file size than an mp3 and condenses factors like tempo, pitch, and duration into a numerical format that a neural network can interpret. Often, unique MIDI configurations are needed when applying neural networks; for instance, MusicBERT uses a custom OctupleMIDI format that is both universal and more efficient than general MIDI formats (2). Similarly, in PopMAG, MUlti-track MIDI (MuMIDI) was created to generate simultaneous tracks that mesh with each other and to reduce sequence length (6). Out of these formats, OctupleMIDI was selected for this project because it is most relevant to the classification task.

MATERIALS AND METHODS

Dataset Curation

Data was sourced from the GitHub repository DeepLearning4Music, which includes folders of MIDI files that exemplify various classical excerpts (13). The following folders were selected as sources of MIDI excerpts: "DeepLearning4Music/data/Classical Archives - The Greats (MIDI)" and "DeepLearning4Music/data/Classical_www.midiworld.com_MIDIRip."

After the data were downloaded locally, they were sorted by composer. Then, using a list of composers organized by time period, individual files were sorted into folders representing the Baroque, Classical, and Romantic eras. Composers represented include Johann Sebastian Bach, George Frideric Handel, Alessandro Scarlatti, Antonio Vivaldi, Dieterich Buxtehude, and Carl Philipp Emanuel Bach for the Baroque period; Joseph Haydn, Ludwig van Beethoven, Muzio Clementi, Wolfgang Amadeus Mozart, and Anton Diabelli for the Classical period; and Georges Bizet, Felix Mendelssohn, Frédéric Chopin, Niccolò Paganini, Robert Schumann, Jean Sibelius, Antonín Dvořák, Pyotr Ilyich Tchaikovsky, Sergei Rachmaninoff, Franz Liszt, Claude Debussy, Otto Nicolai, Max Reger, Jules Massenet, Carl Reinecke, and Nikolai Rimsky-Korsakov for the Romantic period (4).

Next, after MIDI files that could not be processed were removed, the size of the dataset was regulated by reducing each folder down to around 200 samples using random selection. In the end, the Baroque folder had 205 samples (total size 3MB); the Classical folder had 186 samples (total size 9.3MB); and the Romantic folder had 193 samples (total size 7.4MB). Figure 1 illustrates the distributions of MIDI file sizes in the Baroque category (a), the Classical category (b), and the Romantic category (c). Though there were the most Baroque samples, their total size and median size were the least because the excerpts were generally brief individual movements from a shorter piece like a suite or chorale. In contrast, Classical and Romantic examples tended to be movements from longer pieces like sonatas and symphonies.

The number of MIDI tracks per sample in each category was also graphed as shown in Figure 2, with each track representing the sequence of notes played by one instrument in the piece. In general, the distributions of the number of MIDI tracks were skewed right, with a few outliers for pieces with exceptional numbers of instruments.

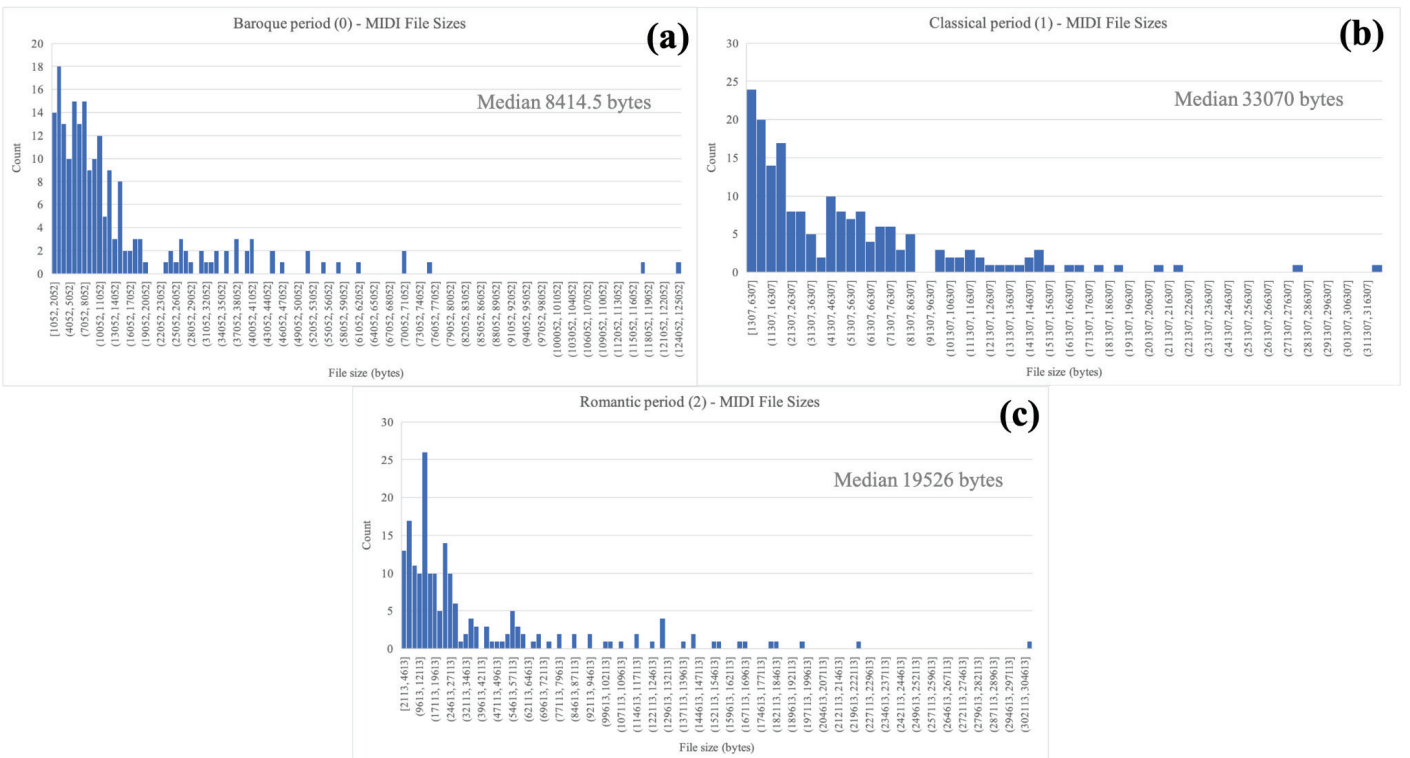

Figure 1. Histograms of MIDI file sizes by period.

Figure 3 illustrates piano roll representations of (a) Antonio Vivaldi's "3rd Movement of Concerto for Violin, Strings and Continuo 'La Stravaganza'" (Baroque), (b) Ludwig van Beethoven's "Piano Sonata no. 15 'Pastoral'" (Classical), and (c) Frédéric Chopin's "Étude Op. 10, No. 5 'Black Keys'" (Romantic). Each green row of white notes represents a MIDI track; from these piano rolls, one can observe some characteristic stylistic features like the

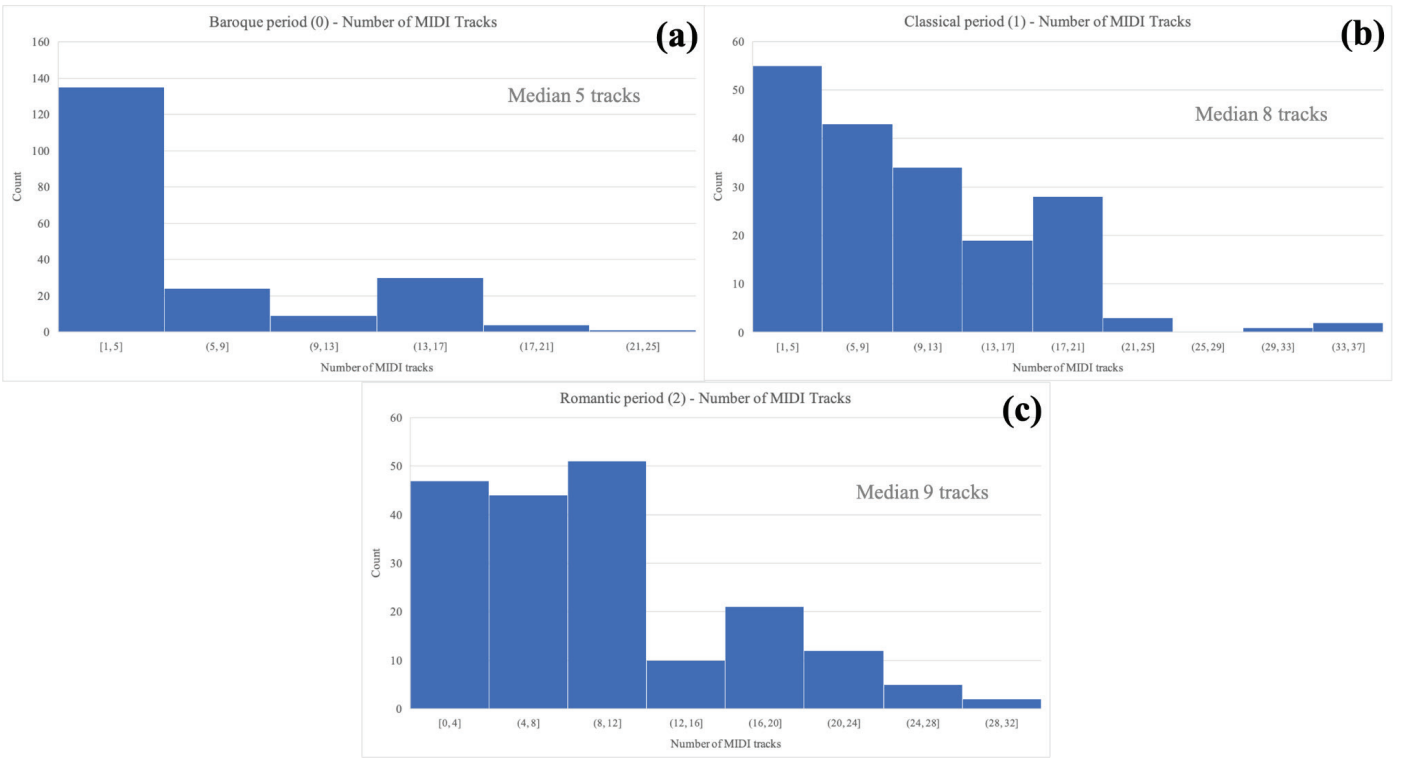

Figure 2. Histograms of number of MIDI tracks per file by period.

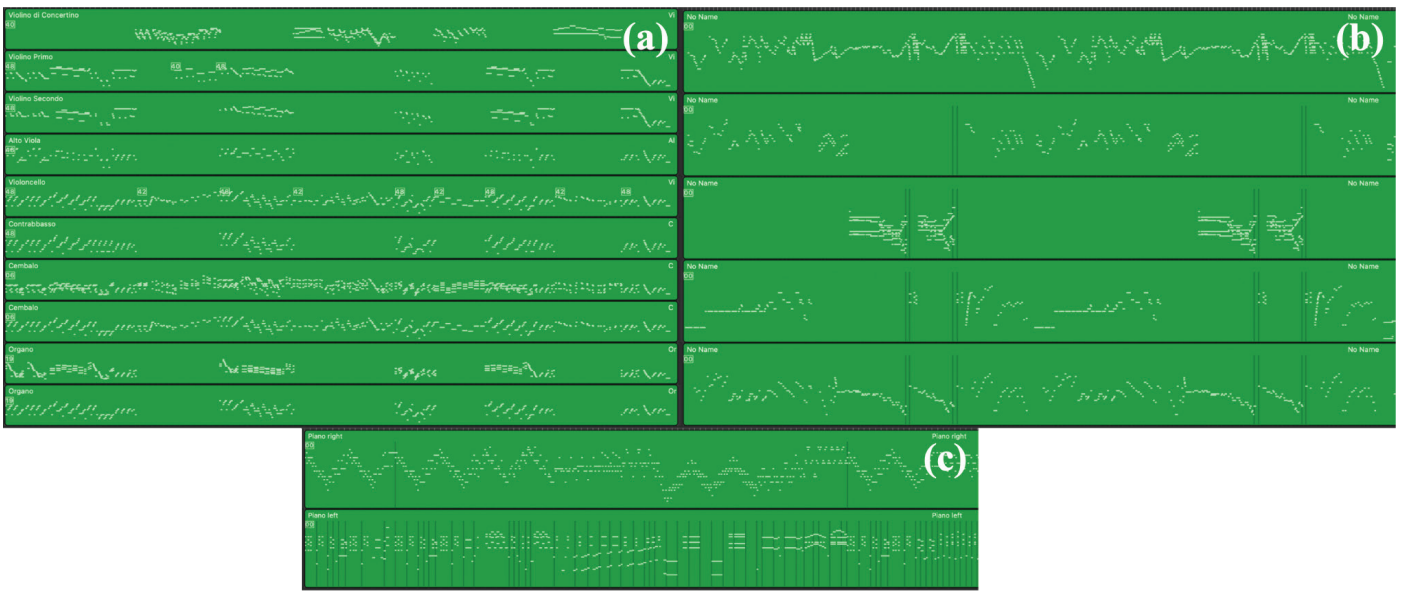

Figure 3. Piano roll representations of MIDI files for Baroque, Classical, and Romantic pieces.

use of complex, counterpoint harmonies in the Baroque piece; the distinctive use of a main melody with different accompaniments in the Classical piece; and the expressive up-and-down movement in the Romantic piece.

Dataset Representation

As shown by the flowchart in Figure 4, after being sorted by category, all MIDIs were converted to the OctupleMIDI format using the process in the Midiformers Google CoLab notebook (12). Each folder containing MIDI samples was traversed and the encoded OctupleMIDI representations were saved individually in a new location. In the OctupleMIDI format, a group of eight numbers numerically represents time signature, tempo, bar, position, instrument, pitch, duration, and velocity of one note; each octuple is counted as one token, or unit of data considered at once, when the model is trained (2). Figure 5 shows the distribution of the number of octuples in each MIDI track for each category.

Then, the OctupleMIDI representations were organized

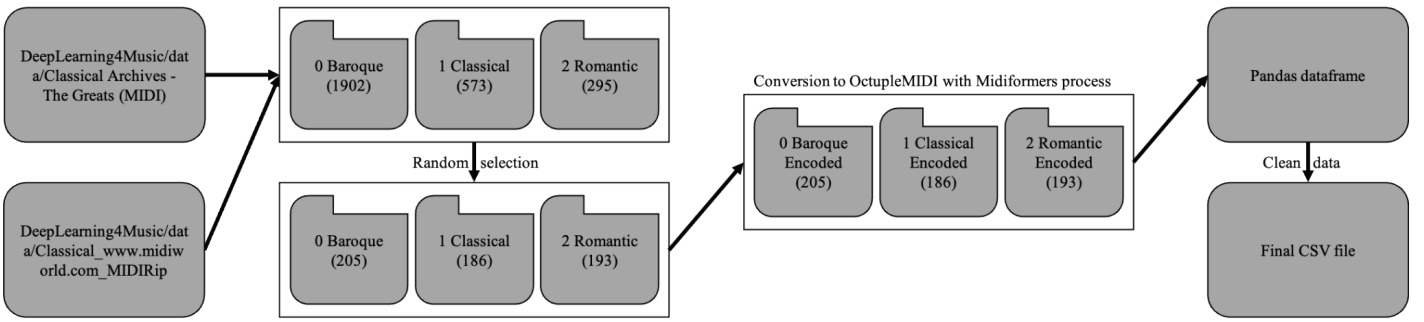

Figure 4. Data processing pipeline from MIDI download to final CSV for training.

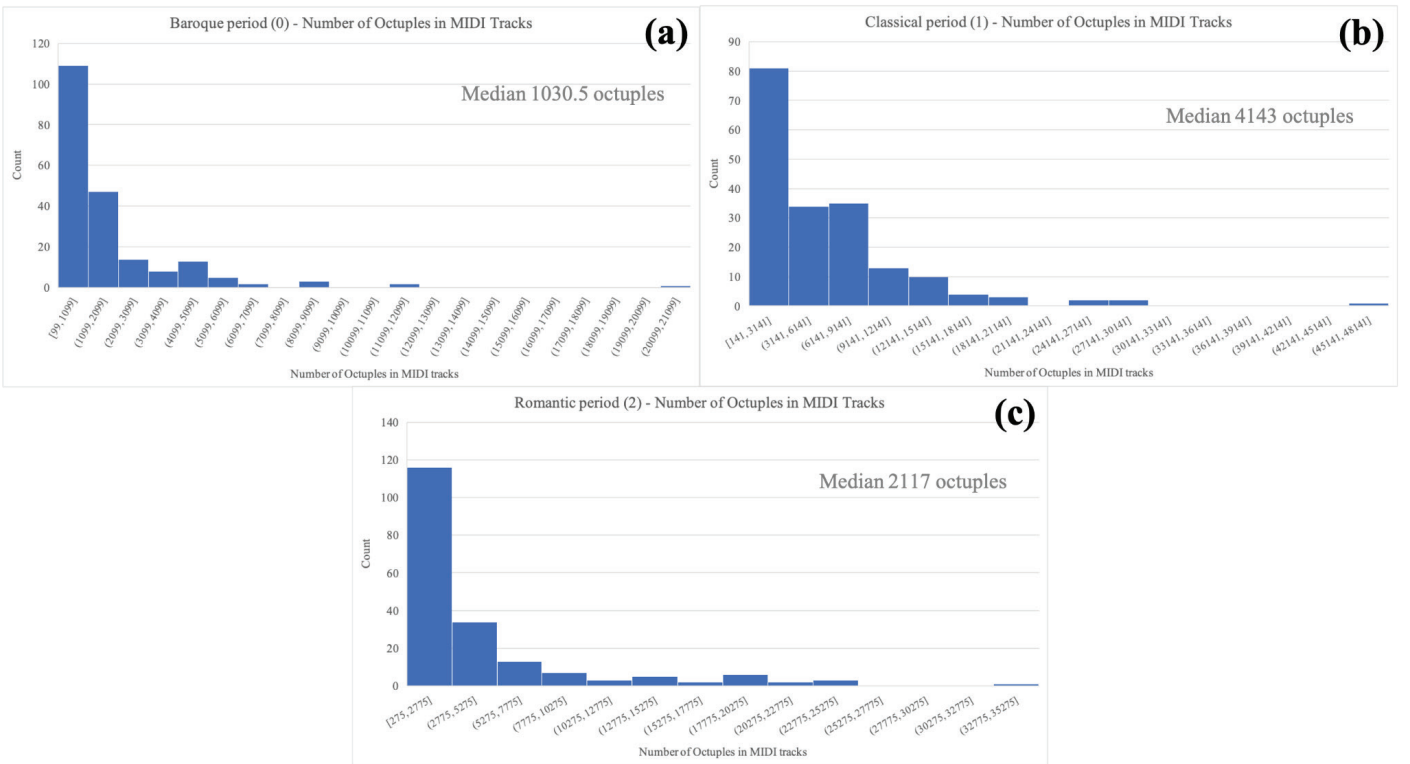

Figure 5. Histogram of number of octuples per MIDI track per file by period.

into a Pandas dataframe according to HuggingFace specifications, with the "text" column containing "sentences" of OctupleMIDI octuples, and the "label" column containing 0, 1, or 2 for Baroque, Classical, or Romantic respectively. This dataframe was then saved as a CSV file for future use. Since the OctupleMIDI representation came with certain parameters represented as np.int8(x) instead of only x, these values were standardized to match the others with find and replace in the CSV.

Preparing for Training

Using a guide to loss functions for Sentence Transformers, BatchSemiHardTripletLoss was initially chosen as a balance between stability and efficiency that also accepted pairs of one "sentence" (the OctupleMIDI representation) and one label, matching the dataset format (14). Later, the loss function (a calculation of the discrepancy between the predicted and actual value) was changed to CrossEntropyLoss from PyTorch since BatchSemiHardTripletLoss was less appropriate for classification tasks such as this one; other Sentence Transformers for classification tasks like SoftmaxLoss were incompatible with pairings of one "sentence" with a label.

The model sentence-transformers/paraphrase-MiniLM-L6-v2 was chosen as the base model since it was the default. A simple classifier head consisting of a fully connected linear layer was added to the base model to complete the classification task.

Data was read from the aforementioned CSV file. Using roughly a 75%-25% training-testing split, 135 examples per label were initially randomly separated as training examples with the rest as testing examples. Training and testing dataloaders were created with a batch size of 8—the number of training samples simultaneously processed before the model updated its weights—due to the limited processing power available. Through experimentation, a batch size of 16 was found to be less effective.

Training Process

An evaluation function was defined to print a confusion matrix as follows, where TP indicates True Positive (correct classification) and FN indicates False Negative (misclassification) as shown in Table 1. The evaluation function also printed the accuracy rate of the model at that stage using the accuracy_score() function.

Training used the machine learning framework PyTorch and the Adam optimizer (an algorithm to adjust weights and parameters to train the model). Each training step first zeroed gradients to ensure parameters were updated independently with each step. Then, a forward pass was conducted through the SentenceTransformer to convert data to embeddings and through the classifier to get the probability of each category being the correct label. Finally, predictions were compared against true labels with the loss function, and weights and parameters were adjusted through backpropagation and an optimizer step. The training loop was established with a number of epochs (complete passes through the training data) that was varied in later tests; at the end of each epoch, the evaluation function was run and a printout of the confusion matrix and accuracy score at that point was produced.

At the end of training, the model was tested on the testing dataset to print a final confusion matrix and accuracy score. In order to maximize accuracy after each trial, the following hyperparameters were modified: loss function, batch size, learning rate (a parameter that decides the amount by which the model adjusts parameters at each step), number of epochs, and training set size.

RESULTS

The training was first conducted with the model sentence-transformers/paraphrase-MiniLM-L6-v2. The loss function BatchSemiHardTripletLoss was used to train for three epochs. However, the accuracy rate was only 0.2216.

In Trial 1, the loss function was changed to

Table 1. Explanation of confusion matrix of predicted and actual music period classifications

	Predicted Baroque	**Predicted Classical**	**Predicted Romantic**
Actual Baroque	TP (Baroque)	FN (Baroque)	FN (Baroque)
Actual Classical	FN (Classical)	TP (Classical)	FN (Classical)
Actual Romantic	FN (Romantic)	FN (Romantic)	TP (Romantic)

CrossEntropyLoss and the number of epochs was increased to 20 with the default batch size 8, learning rate 2e-5, and maximum training set size of 135 samples, yielding an accuracy rate of 0.4716. In Trial 2, an alternative base model sentence-transformers/all-mpnet-base-v2 was tested with a batch size of 16 (other hyperparameters were kept the same as Trial 1), yielding an accuracy rate of 0.4122. Since the accuracy was lower, the base model was changed back to sentence-transformers/paraphrase-MiniLM-L6-v2. Confusion matrices resulting from the configurations in Trial 1 and Trial 2 are illustrated in Table 2, where accurate predictions are bolded.

Next, different values of learning rate were changed: 1e-5, 1e-4, 1e-3, 1e-2, and 1e-1. Keeping epochs (10), batch size (8) and training set size (135) constant, accuracy increased from 1e-5 to 1e-2, then dropped at 1e-1, as shown in Table 3.

In Trial 3, keeping the optimal learning rate of 1e-2, the number of training examples was increased to 175, with batch size still at 8. 20 epochs of training produced the confusion matrix shown in Table 4, where accurate predictions are bolded, yielding an accuracy rate of **0.6250**.

This value corresponds to the values of precision, recall, and F1-score per label shown in Table 5. Using TP/TN/FP/FN to represent true positives/true negatives/false positive/false negatives respectively, precision P is $\frac{TP}{TP + FP}$; recall R is $\frac{TP}{TP + FN}$; and F1-score is the harmonic mean of precision and recall, $\frac{2 \times P \times R}{P+R}$.

Since a higher accuracy was unlikely to be obtained through solely modifying hyperparameters, the Classical and Romantic labels were combined into one category. These two groups were the most frequently misidentified in the confusion matrices. Using the two new categories (Baroque and Classical-Romantic), in trial 4, the model was trained on 20 epochs, learning rate 1e-2 (as through trials of different rates 1e-2 still performed the best), batch size 8, and 175 training examples; this yielded an accuracy rate of 0.9048. In trial 5, a more balanced testing set that

Table 3. Accuracies with different learning rates

Learning Rate	Accuracy
1e-5	0.5
1e-4	0.5284
1e-3	0.5739
1e-2	**0.5795**
1e-1	0.4318

Table 4. Confusion matrix for Trial 3

	Predicted Baroque	Predicted Classical	Predicted Romantic
Actual Baroque	**21**	7	1
Actual Classical	1	**6**	3
Actual Romantic	1	8	**8**

Table 5. Precision, recall, and F1-score per label for Trial 3

Label	Precision	Recall	F1-score
Baroque	0.913	0.7241	**0.8077**
Classical	0.2857	0.6	**0.3871**
Romantic	0.6667	0.4706	**0.5517**

Table 2. Confusion matrices from Trials 1 and 2

	Trial 1				Trial 2		
	Predicted Baroque	Predicted Classical	Predicted Romantic		Predicted Baroque	Predicted Classical	Predicted Romantic
Actual Baroque	**42**	11	16	Actual Baroque	**5**	13	36
Actual Classical	13	**13**	24	Actual Classical	0	**13**	22
Actual Romantic	9	20	**28**	Actual Romantic	0	6	**36**

included fewer Classical-Romantic selections was used, yielding an accuracy rate of 0.8333. The corresponding confusion matrices (accurate predictions bolded) and precision, recall, and F1-scores per label in Trials 4 and 5 are shown in Table 6.

DISCUSSION

Out of the hyperparameter combinations tested, the most effective used the model sentence-transformers/paraphrase-MiniLM-L6-v2, the loss function CrossEntropyLoss, 20 epochs of training, batch size 8, a maximum of 175 training examples, and a learning rate of 1e-2. This set of values allowed a maximum accuracy of 0.6250 with three classes (Table 4), and 0.9048 with two classes (Table 6).

As expected, more training examples led to higher accuracy, as the model had more data to work with and discern trends from. Similarly, more epochs of training led to higher accuracy, as more epochs lent the model more opportunities to learn from the data and adjust parameters accordingly. The increase and decrease in the learning rate could also be expected; up to a certain point, increasing the learning rate will improve accuracy by helping the model update parameters faster and converging to a solution.

However, an overly high learning rate would lead to overfitting—where the model can accurately predict the class that samples in the training data belong to, but cannot generalize these trends to new examples it has not encountered before—or to overshooting the point of the optimal parameters, leading to a suboptimal solution (1).

In relation to the original topic of inquiry, the BERT-adjacent Sentence Transformer model was effective in classifying genres of classical music. Other models like long-short-term memory networks (LSTMs) were also tested, but they proved less effective, only attaining an accuracy rate of 0.7651 with the two categories merged. Fine-tuning a large language model like ChatGPT was also considered; however, because of the large amounts of data once processed in OpenAI's specified format, this kind of fine-tuning would be less practical.

Compared with past studies like MusicBERT which classified music genres excluding classical sub-genres, training Sentence Transformers proved less effective. This discrepancy may be partially because of similarities of all pieces within the general classical label; a statistical analysis of sonatas across all three periods reveals the total number of notes, maximum intervals reached, and number of raised and lowered notes remain consistent across 17th to 19th century music (15). By contrast, modern genres like R&B and rock are much more distinct from each other; modern genres are also significantly different from classical music.

The model consistently distinguished Baroque music from Classical and Romantic music. Even with three categories, the model demonstrated an F1 score of 0.8077 for Baroque music, compared to 0.3871 and 0.5517 for Classical and Romantic respectively (Table 5). When a balanced testing set of Baroque and Classical-Romantic examples were used, more similar F1 scores of 0.8438 and 0.8214 were obtained (Table 6).

This ability to differentiate Baroque music from Classical and Romantic music aligns with trends established by musicologists. Because many of the key features of Baroque music—complex counterpoint and rigid forms like toccata and fugue—were largely replaced by less strict harmonies and new forms like concertos and sonatas in the Classical period, these two genres tend to be easy to differentiate; meanwhile, the Romantic period

Table 6. Confusion matrices and precision, recall, and F1-scores per label for Trials 4 and 5

Trial 4					
	Predicted Baroque	**Predicted Classical-Romantic**	**Precision**	**Recall**	**F1-score**
Actual Baroque	**18**	11	0.6207	0.6207	**0.6207**
Actual Classical-Romantic	11	**191**	0.9455	0.9455	**0.9455**
Trial 5					
	Predicted Baroque	**Predicted Classical-Romantic**	**Precision**	**Recall**	**F1-score**
Actual Baroque	**27**	3	0.7941	0.9	**0.8438**
Actual Classical-Romantic	7	**23**	0.8846	0.7667	**0.8214**

built upon trends present in previous compositions while allowing for looser expressionism, making it more similar to the Classical period (3). There are also more composers in the Classical and Romantic periods whose compositional activity crosses the chronological boundaries between the two periods, providing evidence there may be stylistic overlaps between Classical and Romantic (4).

Further improvements can be made by experimenting with more models under Sentence Transformers; increasing the amount of training data; filtering selected excerpts for pieces most representative of their time period; and eliminating selections that could be ambiguously classified between two eras to a human listener.

CONCLUSION

To alleviate the lack of models focusing on genre classification within classical music, this project attempted to train Sentence Transformers models to sort classical music pieces, converted from MIDI to the OctupleMIDI format, into Baroque, Classical, and Romantic era compositions. With a maximum achieved accuracy score of 0.6250 for 3 eras and 0.9048 for 2 eras, this approach was partially effective in identifying classical music eras. With further improvement, this classification task could prove helpful to musicologists when it is difficult to distinguish styles that exhibit characteristics of similar musical periods manually; to music cataloguing or recommendation services in identifying and tagging classical music genres; and for further investigations into quantitative differences between sub-genres of classical music as opposed to trends identified by the human ear.

ACKNOWLEDGEMENTS

I would like to thank Dr. Husni Almoubayyed from Carnegie Learning for his guidance on this
project.

DECLARATION OF CONFLICT OF INTERESTS

The author declares that there are no conflicts of interest regarding the publication of this article.

REFERENCES

1. Jaiswal S. Multilayer perceptrons in machine learning. 2025. Available from: https://www.datacamp.com/tutorial/multilayer-perceptrons-in-machine-learning (accessed on 2025-04-14).
2. Zeng M, Ren Y, Wang J, Chen H, Zhang X, and Liu J. MusicBERT: Symbolic music understanding with large-scale pre-training. *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. 2021; 2021: 1507–1518. https://doi.org/10.18653/v1/2021.findings-acl.70
3. Dalla Bella S, and Peretz I. Differentiation of classical music requires little learning but rhythm.*Cognition*. 2005; 96 (2): B65–B78. https://doi.org/10.1016/j.cognition.2004.12.005
4. University of Wisconsin La Crosse. Search@UW | Murphy Library. Available from: https://libguides.uwlax.edu/c.php?g=614952&p=4275908 (accessed on 2025-04-14).
5. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, *et al*. Attention is all you need. *Advances in Neural Information Processing Systems*. 2017; 30: 5998–6008.
6. Ren Y, Hu H, Tan Q, Yang J, *et al*. PopMAG: Pop music accompaniment generation. *Proceedings of the 28th ACM International Conference on Multimedia*. 2020; 2020: 1198–1206. https://doi.org/10.1145/3394171.3413721
7. Muller B. BERT 101 – State of the art NLP model explained. Available from: https://huggingface.co/blog/bert-101 (accessed on 2025-04-14).
8. Muller B. Getting started with BERT. Available from: https://colab.research.google.com/drive/1YtTqwkwaqV2n56NC8xerflt95Cjyd4NE (accessed on 2025-04-14).
9. Alammar J. A visual notebook to using BERT for the first time. Available from: https://colab.research.google.com/github/jalammar/jalammar.github.io/blob/master/notebooks/ber t/A_Visual_Notebook_to_Using_BERT_for_the_First_Time.ipynb (accessed on 2025-04-14).
10. Wu Y, Huang S, Engel J, Roberts A, and Sun G. The Chamber Ensemble Generator and Cocochorales dataset. Available from: https://magenta.tensorflow.org/ceg-and-cocochorales (accessed on 2025-04-14).
11. Tripathi A. Midiformers. Available from: https://github.com/tripathiarpan20/midiformers (accessed on 2025-04-14).
12. Tripathi A. MusicBERT mask prediction (latest multiprogram). Available from: https://colab.research.google.com/gist/tripathiarpan20/159ad4330086f4934e5b0a093eb2434b (accessed on 2025-04-14).
13. Almoubayyed H. DeepLearning4Music. Available from: https://github.com/hsnee/DeepLearning4Music (accessed on 2025-04-14).
14. Reimers N, and Gurevych I. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019. Available from: https://arxiv.org/abs/1908.10084
15. Chang L. A comparative statistical analysis of music styles (seventeenth–nineteenth centuries). *Interdisciplinary Science Reviews*. 2020; 45 (4): 581–594. https://doi.org/10.1080/03080188.2020.1849521