**Bach or Not, Classical Composer Classification**

Final Team Project

Group 2:

Carrie Little (clittle@sandiego.edu)

Alexis Lim (alexislim@sandiego.edu)

Ahmad Milad (amilad@sandiego.edu)

Applied Artificial Intelligence,

University of San Diego

AAI-511-03 Neural Networks and Deep Learning

Dr. Mirsardar Esmaeili

08/11/2025

# Bach or Not, Classical Composer Classification

## Abstract

This project investigates the classification of classical music compositions by composer using three deep learning architectures: a Convolutional Neural Network (CNN), a Long Short-Term Memory network (LSTM), and a hybrid CNN-LSTM model. The models are trained to predict one of four composers: Bach, Beethoven, Chopin, or Mozart, using symbolic music data extracted from MIDI files. We evaluate model performance based on accuracy and classification metrics, and describe the architecture and hyperparameter choices for each model. Results indicate that the LSTM model captures long-range musical structure well for Bach, while the CNN model is efficient in identifying local motifs. The hybrid model offers a more balanced and generalized approach across composers.

## Introduction

Musical compositions by classical composers exhibit stylistic signatures that machine learning algorithms can capture. Deep learning offers the opportunity to automatically learn high-level representations from musical sequences, thereby eliminating the need for manual feature engineering. This project compares three deep learning approaches: CNN, LSTM, and a CNN-LSTM hybrid, for composer classification. We analyze how different architectures impact performance and how model hyperparameters and preprocessing choices contribute to the learning of musical patterns.

## Dataset Description

The dataset is derived from Kaggle's "midi-classic-music" collection and consists of MIDI files representing compositions from multiple classical composers. For this project, we selected four composers with distinguishable styles: Johann Sebastian Bach, Ludwig van Beethoven, Frédéric Chopin, and Wolfgang Amadeus Mozart. MIDI files were parsed using the pretty\_midi library to extract note sequences, pitch data, durations, and tempo.

## Methodology

### Data Preprocessing

All MIDI files were parsed and converted into sequences of note pitches using pretty\_midi. These sequences were tokenized into integers and padded to a uniform length. For CNN and CNN-LSTM models, a channel dimension was added. Label encoding was applied to the composer names, and the dataset was split into training and test sets using stratified sampling to preserve class distribution.

### Feature Extraction

Two categories of features were extracted:

* Sequence-based features: Raw pitch sequences tokenized and padded.
* Statistical features: Note count, pitch and duration statistics, tempo averages, and key signature.

These features were used primarily to support understanding of distributional properties and were not included directly in the model input.

### CNN Architecture

#### The CNN model includes:

* Input layer: padded note sequence with one-hot representation.
* Conv1D layer (64 filters, kernel size 3, ReLU activation): captures local note patterns.
* MaxPooling1D (pool size 2): downsampling.
* Conv1D layer (128 filters, kernel size 3): higher-level motif recognition.
* GlobalMaxPooling1D: reduces to 1D feature vector.
* Dense layer (64 units, ReLU): nonlinear transformation.
* Dropout (0.3): regularization.
* Output layer (softmax): class probabilities.

Hyperparameters tuned:

* Kernel sizes: 3 to 5
* Dropout rate: 0.2 to 0.4
* Optimizer: Adam with default learning rate

### LSTM Architecture

#### The LSTM model includes:

* Embedding layer (input\_dim = vocab size, output\_dim = 128): maps note indices to dense vectors.
* LSTM layer (128 units): learns temporal dependencies.
* Dense layer (64 units, ReLU)
* Dropout (0.3)
* Output layer (softmax)

Hyperparameters tuned:

* Embedding size: 64 and 128
* LSTM units: 64 and 128
* Batch size: 32
* Epochs: up to 20 with early stopping

### Hybrid CNN-LSTM Architecture

This architecture combines convolutional pattern extraction with temporal modeling:

* Input layer and reshape to (sequence\_length, 1)
* Conv1D (64 filters, kernel size 3)
* MaxPooling1D (pool size 2)
* Conv1D (128 filters)
* Bidirectional LSTM (128 units): captures bidirectional temporal dependencies
* LSTM (64 units): deeper sequence modeling
* Dense (64 units, ReLU), Dropout (0.4)
* Output (softmax)

Hyperparameters tuned:

* Convolution filter size and depth
* LSTM layer depth and units
* Learning rate and patience for ReduceLROnPlateau

### Results & Evaluation

#### CNN Model

* Accuracy: 70%
* Bach F1-score: 0.85
* Macro Avg F1-score: 0.50
* Observation: Strong pattern recognition, limited temporal context.

#### LSTM Model

* Accuracy: 70%
* Bach F1-score: 0.87
* Macro Avg F1-score: 0.49
* Observation: Strong performance on Bach, weaker on other composers due to overfitting.

#### CNN-LSTM Model

* Accuracy: 68%
* Bach F1-score: 0.84
* Macro Avg F1-score: 0.46
* Observation: Better class balance and generalization, though slightly lower peak accuracy.

## Discussion

Each model exhibits strengths and limitations. The CNN model is effective at recognizing short motifs and themes but lacks temporal memory. The LSTM model captures long-term structure well but struggles with subtle stylistic variations. The CNN-LSTM hybrid architecture offers a trade-off between local pattern recognition and temporal context, thereby improving the balance across all composers. Despite hyperparameter tuning, all models exhibit a performance bias toward Bach, likely due to the dataset's distribution or stylistic dominance.

## Conclusion & Future Work

This study demonstrates that deep learning architectures can classify classical composers with moderate success using symbolic music data. The LSTM model excels in modeling temporal structure, while the CNN model efficiently extracts motifs. The CNN-LSTM model offers a balanced approach. Future improvements could include data augmentation, Transformer-based architectures, and the incorporation of additional musical dimensions, such as harmony, rhythm, and dynamics. Larger and more balanced datasets would further enhance generalization.

## References

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

Kaggle (n.d.). MIDI Classical Music Dataset. https://www.kaggle.com/datasets/blanderbuss/midi-classic-music

Pretty MIDI Library: https://github.com/craffel/pretty-midi