# Classification of Classical Music Composers Using Deep Learning Models

Andrew Roher, Aryaz Zomorodi, and Bilal Najar

Shiley-Marcos School of Engineering, University of San Diego

**Abstract**

This project aims to classify classical music compositions by composer using deep learning techniques. We utilized two types of neural networks, Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks, to process and analyze MIDI files representing compositions from various classical composers. The project involved comprehensive data pre-processing, feature extraction, and model training to achieve high classification accuracy. This report details the methodology, data pre-processing steps, feature extraction techniques, model architecture, and training process for reproducibility and future reference.

*Keywords*: Deep Learning, CNN, LSTM, MIDI, Music Classification, Feature Extraction, Data Augmentation

## Methodology

### Data Selection

This study aims to design and evaluate a model for classifying classical music compositions by composers using deep learning techniques (LSTM and CNN). Utilizing the *'midi_classic_music*' dataset from Kaggle, the research seeks to showcase how artificial intelligence (AI) can accurately detect and classify classical music compositions by composer.

### Data Acquisition and Preparation

This step involved setting up libraries such as matplotlib, sklearn, and tensorflow.keras on Google Colab. The MIDI files dataset is categorized by four specific composers:

- Bach
- Beethoven
- Chopin
- Mozart

The dataset was obtained from the Kaggle website and uploaded to our GitHub repository.

### Data Preprocessing

The pre-processing stage converts the musical scores into a format suitable for deep learning models. This involves:

- MIDI File Handling: Fetching MIDI file paths from specified directories and loading them into *pretty_midi* objects.
- Feature Extraction: Extracting note-level and complex features from each MIDI file, including:
    - Instrument program number
    - Whether the instrument is a drum
    - Note start and end times

        - Note pitch and velocity
        - Tempo changes and statistics (mean, min, max, variance)
        - Key signature changes converted to numeric format
        - Time signature changes converted to numeric format
            - The extracted features are compiled into a dataset with columns representing each attribute.

- Data Augmentation: Applying data augmentation techniques to enhance the dataset. This involves shifting and stretching pitch sequences to create new, varied samples.
- Normalization: Normalizing the sequences to a common scale for consistent input to the models.
- Padding Sequences: Padding the sequences to a uniform length to ensure consistency in model input dimensions.
- Encoding Categorical Data: Converting categorical labels (composer names) to numerical format using label encoding.
- Saving Processed Data: Saving the preprocessed features to a CSV file for model training.

**Feature Extraction**

In this notebook, the feature extraction process involved using the *pretty_midi* library to extract various musical attributes from MIDI files. The *extract_features* function was implemented to process individual MIDI files, extracting note-level features such as instrument program, drum status, start time, end time, pitch, and velocity. Additionally, it extracted higher-level features including tempo statistics (mean, min, max, variance), key signature information, and time signature details. The *extract_features_from_directory* function was then used to apply this extraction process across an entire dataset of MIDI files, organizing the data by composer. The extracted features were compiled into a pandas DataFrame and saved as a CSV file for further processing.

**Model Building**

The model building phase focused on creating two types of neural networks: a Convolutional Neural Network (CNN) and a Long Short-Term Memory (LSTM) network. CNNs excel at detecting local patterns and hierarchical features in data, while LSTMs are particularly effective at capturing long-term dependencies in sequential data, making them both ideal for analyzing musical structures and temporal patterns in MIDI files (Lilhore et al, 2023). The CNN model was constructed using Conv1D layers, MaxPooling1D layers, and Dense layers, with dropout for regularization. The LSTM model comprised two LSTM layers followed by Dense layers, also incorporating dropout. Both models were compiled using the Adam optimizer and sparse categorical crossentropy loss function. The input data was reshaped to fit the requirements of each model type (3D shape for both CNN and LSTM). The models were then trained on the extracted features, with the notebook showing the training process for ten epochs. Evaluation metrics including accuracy, precision, and recall were calculated for both models, and confusion matrices were generated to assess their performance.

**Model Evaluation**

The performance of the deep learning models, including both the Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) network, was evaluated using key metrics: accuracy, precision, and recall. These metrics are critical for assessing how well the models generalize to unseen data and how effectively they can classify musical scores according to their composers.

***CNN Model Evaluation***

- **Accuracy**: The CNN model achieved an accuracy of 98.51%, indicating that it correctly classified the vast majority of the musical scores in the test dataset.
- **Precision**: The CNN model achieved a weighted precision of 98.51%. This high precision indicates that the model is highly effective in minimizing false positives across all classes.

- **Recall**: The recall for the CNN model was also 98.51%, reflecting the model's ability to capture almost all relevant instances of each composer, with minimal false negatives.

These metrics were calculated using the *classification_report* and *confusion_matrix* functions from the sklearn.metrics module (Pedregosa et al., 2011). The confusion matrix further confirmed the model's strong performance by showing that misclassifications were rare and evenly distributed across classes.

***LSTM Model Evaluation***

- **Accuracy**: The LSTM model achieved an accuracy of 98.52%, slightly outperforming the CNN model. This indicates that the LSTM model was equally effective at generalizing from the training data to new, unseen data.
- **Precision**: The weighted precision of the LSTM model was 98.53%, which is slightly higher than the CNN model's precision. This suggests that the LSTM model may be better at reducing false positives in this context.
- **Recall**: The LSTM model's recall was 98.52%, again slightly higher than that of the CNN model. This indicates that the LSTM model was very effective at capturing relevant instances and minimizing false negatives.

Overall, both models exhibited excellent performance, with only marginal differences between them. The high accuracy, precision, and recall across both models underscore their robustness in classifying musical scores by composer.

**Model Optimization**

The models were optimized through a combination of manual tuning and regularization techniques to achieve high performance while avoiding overfitting.

***Hyperparameter Tuning***

While the notebook did not explicitly utilize automated hyperparameter tuning tools such as Keras Tuner, hyperparameters such as the number of layers, number of units in each layer, dropout rates,

and learning rates were manually adjusted to optimize model performance. This tuning helped to balance model complexity with the ability to generalize well to new data.

### *Regularization Techniques*

Both the CNN and LSTM models incorporated dropout layers, which randomly disable a fraction of neurons during training to prevent the models from becoming overly reliant on specific pathways (Srivastava et al., 2014). This technique is crucial in reducing the risk of overfitting, especially when dealing with complex models and large datasets.

## Conclusion

Both the CNN and LSTM models demonstrated strong performance with high accuracy, precision, and recall. The LSTM model showed a slight edge in precision and recall, suggesting it might be slightly better suited to this specific classification task. The combination of careful hyperparameter tuning and regularization techniques was key to optimizing these models and achieving robust results. For future work, the implementation of early stopping and more extensive hyperparameter tuning could further refine the models' performance.

**References**

Lilhore, U. K., Dalal, S., Faujdar, N., Margala, M., Chakrabarti, P., Chakrabarti, T., Simaiya, S., Kumar, P., Thangaraju, P., & Velmurugan, H. (2023, September 5). *Hybrid CNN-LSTM model with efficient hyperparameter tuning for prediction of parkinson's disease*. Nature News. https://www.nature.com/articles/s41598-023-41314-y

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825-2830.

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*, 1929-1958.