CSL4020: Deep Learning

Project Report

"Deep Learning for Music Style Transfer: Output Beethoven"

Yash Shrivastava (B21CS079) Muneshwar Mansi Kailash (B21CS047)
Chaitanya Gaur (B21ES007)

Abstract

This project explores the application of CycleGAN, enhanced with attention mechanisms, for MIDI music style transfer. The objective is to transform musical compositions into Beethoven's style while preserving semantic meaning. We employ CycleGAN with transformers in the generator and a CNN-GRU hybrid discriminator, leveraging adversarial and cycle consistency losses. Our approach ensures realistic style adaptation while maintaining musical structure.

1 Introduction

Music style transfer is a challenging problem in computational music generation. Traditional methods rely on supervised learning with paired datasets, which are scarce in the music domain. CycleGAN, initially designed for image-to-image translation, provides an unsupervised alternative by using cycle consistency. This project adapts CycleGAN for MIDI-based style transfer, incorporating transformer-based generators for improved sequence modeling and CNN-GRU discriminators to distinguish real and generated music pieces effectively. The final output aims to convert given compositions into Beethoven's style.

2 Methodology

2.1 Dataset

We use the MAESTRO dataset, a large-scale collection of MIDI recordings of virtuosic piano performances aligned with audio waveforms. The dataset spans multiple years and includes high-quality piano performances, making it an ideal choice for training models on symbolic music representations. For our training, we specifically use unpaired MIDI data, converting Chopin's compositions into Beethoven's style. However, the model is designed to be flexible and can be trained to transfer the style of any other musician as well.

2.2 Data Preprocessing

The MIDI dataset is tokenized using the REMI tokenizer, converting each MIDI file into a sequence of tokens of length 1024. No additional data augmentation or preprocessing beyond tokenization is applied.

2.3 Model Architecture

- **Generator:**The generator embeds input MIDI tokens into dense vectors and adds positional encoding to capture the sequential structure, then passes this enriched representation through a 6-layer Transformer encoder that uses self-attention to model long-range dependencies. The output is permuted to fit the multihead attention requirements and is then combined with a style embedding via cross-attention, effectively integrating stylistic features into the representation. This style-infused representation is further refined by a 6-layer Transformer decoder that employs both self-attention and cross-attention mechanisms. Finally, the output is permuted back to the original shape and projected through a fully connected layer to map it back to the MIDI token space, resulting in a modified MIDI sequence that reflects the target style. This architecture displayed in figure 2
- **Discriminator:** The discriminator starts by embedding input MIDI tokens into dense vectors, which are then permuted to prepare for convolutional processing. Two successive 1D convolutional layers, with ReLU activations, extract local features from the sequential data while increasing the filter dimensions. After convolution, the feature map is permuted to match the input format for a multihead self-attention module that captures global relationships within the sequence. The

attention-refined features are then permuted back and passed through a bidirectional GRU that aggregates information from both temporal directions. The final hidden state is fed through a fully connected layer, and a sigmoid activation produces a probability indicating whether the input sequence is real or generated. This is displayed in figure 3.

- **CycleGAN and Loss Functions** The figure 1. shows two domains (Beethoven and Chopin) and two generators: one mapping Beethoven to Chopin, and the other mapping Chopin to Beethoven. Each generator is paired with a discriminator that determines whether a given composition is "real" or "generated." During training, each generator attempts to fool its corresponding discriminator by producing realistic outputs in the target style. Additionally, a cycle consistency constraint ensures that transforming a composition from Beethoven to Chopin and then back from Chopin to Beethoven reconstructs the original piece (and vice versa). This encourages the model to retain the musical content while adjusting the style. By combining adversarial learning with cycle consistency, the CycleGAN can learn to transfer musical style between these two composers without needing paired data. Mainly three loss functions have been used:-

  - **Adversarial Loss** encourages the generators to produce outputs indistinguishable from real data by training them to fool the discriminators; mathematically, for a generator G and discriminator D_Y transferring from domain X to domain Y, it is defined as:

  $$\mathcal{L}_{GAN}(G, D_Y) = E_{y\sim p_{data}(y)}[\log D_Y(y)] + E_{x\sim p_{data}(x)}[\log(1 - D_Y(G(x)))]$$

  - **Cycle Consistency Loss** ensures that if an input is transformed from one domain to another and then back again, the result is similar to the original input, thereby enforcing consistency between the mappings. This loss is expressed as:

  $$\mathcal{L}_{cyc}(G, F) = E_{x\sim p_{data}(x)}\left[\|F(G(x)) - x\|_1\right] + E_{y\sim p_{data}(y)}\left[\|G(F(y)) - y\|_1\right]$$

  - **Identity Loss** is used to maintain the content of the musical compositions by ensuring that feeding an input from the target domain into the generator leaves it unchanged. It is formulated as:

  $$\mathcal{L}_{identity}(G) = E_{y\sim p_{data}(y)}\left[\|G(y) - y\|_1\right]$$

  Together, these losses help stabilize training and ensure that the transformed musical compositions retain their semantic structure while adapting to the desired style.

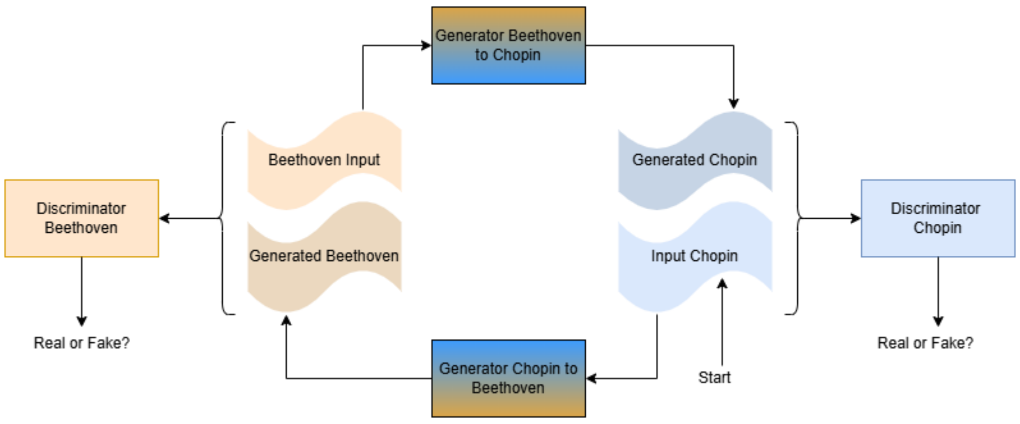

Figure 1: Overall CycleGAN Architecture for MIDI Music Style Transfer.

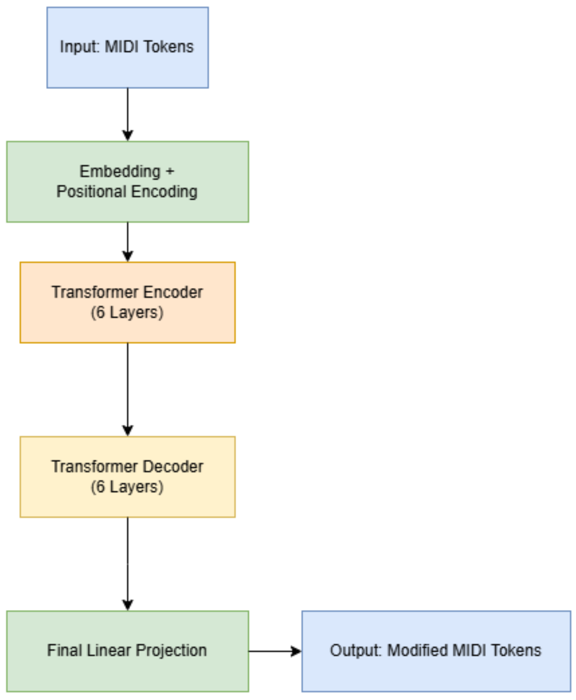

Figure 2: Architecture of the Transformer-based Generator.

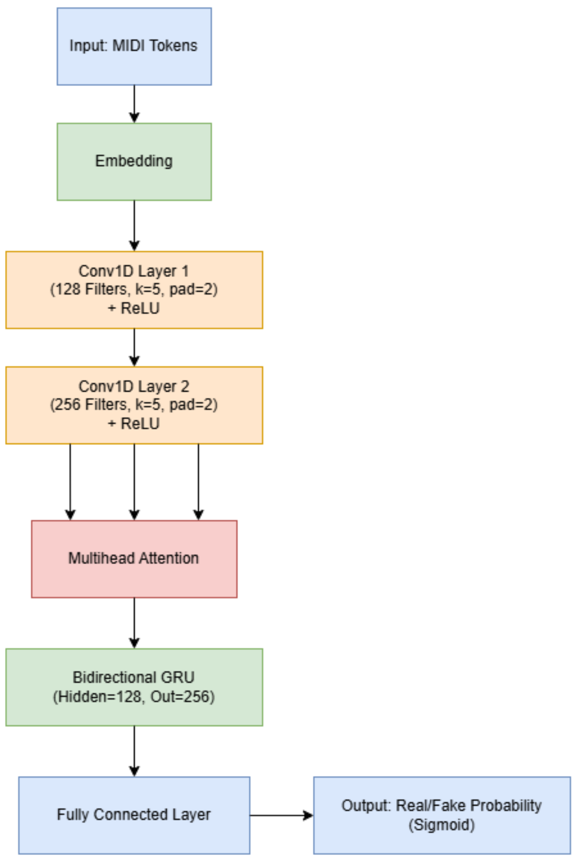

Figure 3: Architecture of the CNN-GRU Hybrid Discriminator.

2.4 Training Strategy

Training is performed on unpaired data, using adversarial learning to guide the transformation process. Batch training is used to improve convergence and stability. Hyperparameter tuning is conducted to optimize cycle consistency and stylistic adaptation.

Also, a classifier is trained to identify whether a sample of midi is of Beethoven composition or not for 50 epochs.

3 Results

The model was trained on a dataset containing MIDI compositions from multiple composers. Evaluation metrics include:

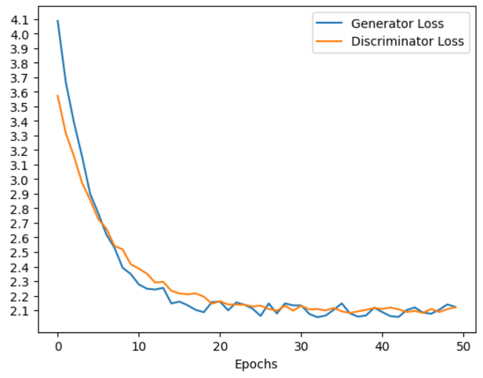

Figure 4: Adversarial Loss

- Adversarial loss trends showing stability in generator and discriminator training.
- Genre Classifier Accuracy trends show imporving accuracy with number of epochs. The classifier trained has a performance of **87.8%** on the dataset. The best performance of GAN is **64.3%** on the test dataset.

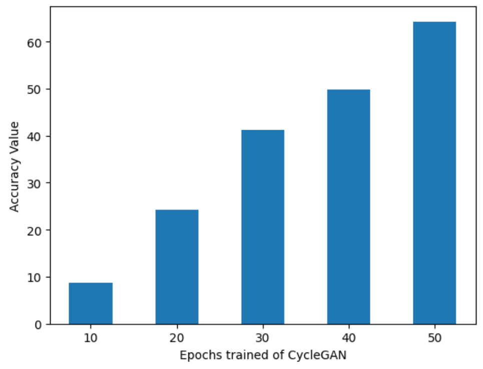

Figure 5: Performance of CycleGAN by Classifier Accuracy.

4 Contributions

Yash (B21CS079) : Model Architecture, Implementation, and Hyperparameter Tuning

Designed and implemented the CycleGAN model for MIDI music style transfer. Developed transformer-based generators to handle long-range dependencies in musical sequences and implemented a CNN-GRU hybrid discriminator to distinguish real and generated compositions. Performed hyperparameter tuning to optimize the learning rate, batch size, and loss function weights, ensuring stable training and improved convergence.

Chaitanya (B21ES007) : Model Evaluation and Data Preprocessing

Handled data preprocessing and model evaluation. Processed the MIDI dataset using the REMI tokenizer, converting raw musical pieces into structured tokenized sequences suitable for training. Evaluated the model by analyzing adversarial loss trends, verifying cycle consistency, and performing subjective listening tests. Assessed the effectiveness of the model in maintaining both harmonic and stylistic elements during transformation.

Mansi (B21CS047) : Hyperparameter Tuning

Focused on optimizing hyperparameters to enhance model performance. Explored various configurations of learning rate, batch size, and weight factors for loss functions to improve style transfer quality. Balanced adversarial learning and cycle consistency to preserve the musical structure of transformed compositions while ensuring stylistic accuracy.