Manny Nkrumah

Dr. Forouraghi

CSC 330

12/6/24

Bach Generation Report

The model is trained on MIDI files containing Johann Sebastian Bach's Cello Suits. Then the files are parsed into sequences of notes and durations to somewhat mimic the underlying patterns and styles in the input music data. The final output is generated MIDI files, reflecting the learned musical patterns. In terms of how the data was prepared, The MIDI files were parsed using the music21 library. During this phase, notes and durations were extracted from the MIDI files. Invalid entries, are marked as "[UNK]" or missing values, then were replaced with rest for notes, 0.25 for durations. In addition, fractional durations were converted into float values for compatibility with the music21 library. Next, the dataset was then tokenized into integer representations using TensorFlow's TextVectorization layer. This step converts sequences into numerical data that the model can process.

Furthermore, a Transformer-based architecture was implemented with the following components: Token and Position Embedding Layers which embeds notes and durations into high-dimensional spaces and adds positional information to encode sequence order. In relation, the transformer block consisting of multi-head self-attention layers, Feed-forward neural networks, layer normalization and dropout for regularization. The dataset was divided into input-

output pairs, where the model learned to predict the next note and duration based on previous sequences.

The training process incorporates loss function: Sparse categorical cross-entropy for both note and duration predictions, Adam optimizer for efficient gradient descent, and 100 epochs over the dataset. The output layers utilizes softmax activation to predict the next note and duration. It is important to note that the model accepts two inputs: sequences of notes and durations. The embeddings of these inputs are concatenated and passed through the Transformer block, followed by two parallel dense layers to output predictions for the next note and duration. The MusicGenerator method generates MIDI files at the end of each epoch using the trained model.

Some challenges that were navigated were notes that the system deemed invalid which caused the system to crash. As a result, I replaced them with quarter rests. In addition, parsing fractional durations caused errors so I utilized Python’s fractions.Fraction module. Finally, the model occasionally produced repetitive patterns. However, regularization through dropout and temperature alterings helped to mitigate overfitting. In terms of the generated music, I considered it a reasonable attempt at emulating Bach’s style, especially in the dynamic changes. However, it falls extremely short of fully capturing the depth, harmonic richness, and contrapuntal precision that define Bach's compositions. Maybe with some more training it could improve.