<u>Generating tonal piano music using machine learning – Deliverable 2</u>

Problem Statement

The goal of my project will be to train a model to generate tonal piano music and store it in MIDI files, so that it can then be played back or converted to a score. I chose this project because one of my passions is composing music and playing the piano, and I've recently become fascinated with music generated by Artificial Intelligence, so I wanted to see if I could train my own model to generate tonal piano music.

Dataset

I had originally planned on using the MAESTRO Dataset, available here, https://magenta.tensorflow.org/datasets/maestro, containing over 200 hours of piano music in MIDI format, but I realized after further research that this dataset contains MIDI files generated from human performance. This means that the durations and timings of the notes are affected by the interpretation of the performer, and so the durations of the notes aren't standardized to the durations of the notes for which musical notation exists. Since my goal was to generate MIDI files that could be converted into scores, I had to find a dataset in which the MIDI files are generated from actual musical scores. I ended up finding this dataset, which contains MIDI files for classical piano music generated from musical scores: http://www.piano-midi.de/midi_files.htm.

I might end up only taking a subset of this dataset for my training, so that the model trains on pieces which are all in the same musical style, or supplementing it with additional data if I find that it isn't enough.

Preprocessing

I preprocessed the MIDI data using the Mido library for Python. This library allowed me to turn the MIDI files into objects containing tracks, with each track containing messages that have a type, and, if that type is "note_on", a time (integer), velocity (integer), and pitch (integer). The time represents the time since the last note was played that this notes should be played, the velocity represents the volume of the note, and the pitch represents the distance in semitones from the lowest possible note (0), with higher integers representing higher pitches. I first had to merge the tracks in the file, since each file had one track for the right hand and one track for the left hand, then had to remove all messages that weren't of type "note_on", then turn those messages into pairs of integers representing only the pitch and time of each note, since those are the only characteristics that were important for my purposes.

Model

After some research, I have decided to implement my model using Keras, due to its ease of use and speed. My model will take a (pitch, time) integer pair as input and predict the (pitch, time) integer pair that follows it. It will then take the (pitch, time) it predicted as input and predict the next (pitch, time), while remembering all the previous (pitch, time) integer pairs it received as input

The first layer in my model will be an LSTM layer. This will allow the model to remember all the previous notes that were used as inputs when deciding the current note to predict, because, in a piece of music, a note doesn't just depend on the note that immediately precedes it, but on all the notes that preceded it in the piece.

The next layer will be a dense layer containing as many nodes as there are possible (pitch, time) integer pairs in my dataset. The (pitch, time) pair with the highest probability will then be selected as the predicted next note in the piece of music.

If possible, I will try to add an attention layer, as my research indicates that this type of layer if very useful in remembering long sequences of inputs, which is exactly what I want in this case.

I might also add a second LSTM layer, with a different numbers of timesteps, so that the first one picks up on shorter patterns in the music, and the second one picks up on the larger structure of the music.