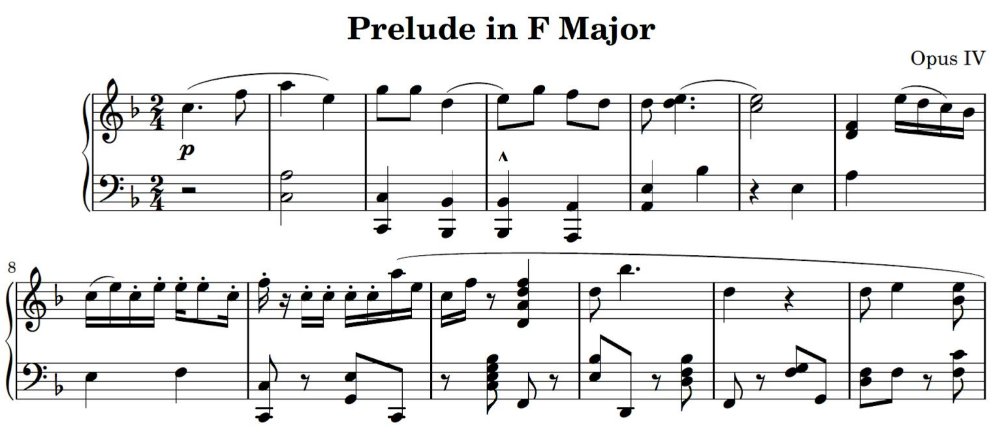
ART AND MACHINE LEARNING
CMU 2019 SPRING
FINAL PROJECT

Polyphonic Music Score Generation



Claire Yuan (chenxiy), Caroline Wu (czw)
https://github.com/claireyuan/ArtML-final-project

Concept:
In this project, we generate polyphonic music scores in the Lilypond format. Lilypond is a music engraving program which uses a Latex-like syntax to encode musical scores [1]. We found a previous work by Seth Colbert-Pollack at Kenyon College attempting to create something similar, but he was only able to generate snippets of monophonic music [3].

Data:
We use a dataset of classical piano Lilypond files from Mutopia [2]. Since there are many ways to encode a music score, standardizing the data was a huge challenge. We ended up cleaning the data both manually and by script. First, we attempt to standardize the length of each training example. In Lilypond, the right and left hand staves are encoded separately. To allow our model to remember both, we shorten our training examples to roughly eight-measure snippets. We do this manually, trying to pick out a sensible musical phrase, and using a script to count bar lines. We also shorten the number of tokens per training example by tokenizing by command instead of by character. To standardize the data even further, we transpose all snippets to be in C major or A minor. We also convert each snippet to use absolute instead of relative pitch.

Model and Training:
We use a text generation recurrent model, experimenting with both LSTM and GRU cells. Due to the scarcity of data, we were forced to train relatively small models, which made it harder to capture the long-term dependencies needed to generate both right and left hand staves. Because we had a smaller quantity of higher-quality manually-cleaned data and a larger quantity of lower-quality script-cleaned data, we began by training on all of the data, and fine-tuned on the higher-quality data. The idea was that the model would learn structure on the combined data, and musicality on the manually-cleaned data. We modified the Pytorch language model example code [4] to allow for this continued training. The fine-tuning did not end up making a noticeable difference.

Results:
Many of our generated snippets were not immediately compilable so we did some manual post-processing to fix syntax and rhythm. Then, because our generated snippets were generally very short, we combined our best snippets to form larger pieces of music. See Appendix for previews of the pieces we generated. See our github for the full music scores and mp3 files.

Reflection:
We were successfully able to generate several pieces of polyphonic music. However, the results generated by our model required a lot of post-processing to be presentable. We were constrained by the amount of data and the effort required to clean the data. Given more time, we would be able to process more data to improve our models.

Reference:
[1] LilyPond... music notation for everyone. (n.d.). Retrieved from http://lilypond.org/index.html
[2] Contributors, M. P. (n.d.). Retrieved from https://www.mutopiaproject.org/index.html

[3] Colbert-Pollack, S. (n.d.). RNN monophonic sheet music generation with LilyPond. Retrieved from https://digital.kenyon.edu/dh_iphs_ai/2/
[4] https://github.com/pytorch/examples/tree/master/word_language_model

Appendix. Preview of Final Results.

Sonatina in C Major

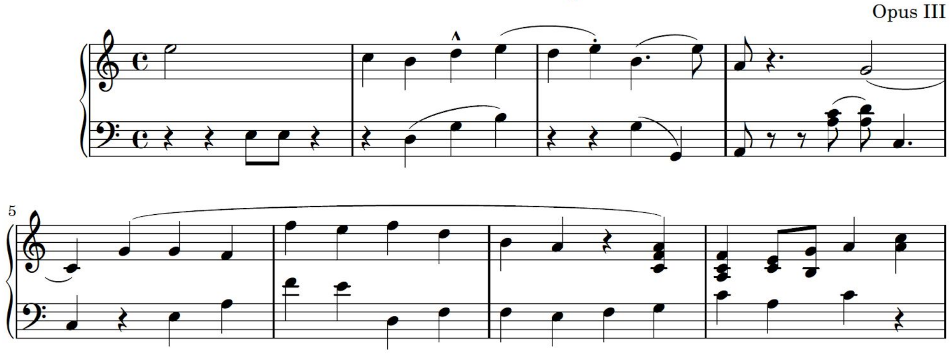
Opus I

Waltz in A Minor

Opus II

Etude in C Major

Opus III



Prelude in F Major

Opus IV

Pavane in A Minor

Opus V

Promenade in G Major

Opus VI