Andy Van Heuit, Trevor Nunn, David Olson
Capstone 2017

Chorale Generator Project Proposal

1. Overview

Our Capstone project entails creating software that can generate chorales. A chorale is a musical composition consisting of a simple tune, and harmonies on it. Our plan is to use a neural network and train on a set of 100 existing chorales by Johann Sebastian Bach. We would then like to broaden this scope and train on significantly more and varied chorales. Extending that, our software will also be able to generate a long chorale off a short, 8-16 bar user-inputted melody that the neural network remembers as its training. While we anticipate the neural network and training conditions with input to take the bulk of our time, we would like to displaying the finished composition in standard sheet music form or at the very least some visual format, as well as a mini-report detailing what logic the program followed to create the piece. The hope for this mini-report is to provide snapshots of the generated chorales as the neural network trains and let us see how the network is learning.

2. Challenges

Some of the hardships we will face include:

- We have a solid open source data set found at UCI machine learning repository, which provides note, pitch, key signatures, etc. This has all the information we think we will need to put into the neural network, however once we provided this to the network and it learns from what we provided, we're not sure how to get this information back into a readable file type. Aside from the neural network itself, we think this might be the biggest hurdle.
- To train on other types of music we would need to get events lists of that music like UCI is providing, or find a way to pull these details from
- Our current plan to tackle the above problems is to use the jMusic library, which should allow us to break down common music files like MIDI into note-files and vice versa, but don't currently have a fall back plan.
- We're pretty sure that we can use a feedforward neural network with our current understanding, but slightly confused on how our proposed user input along with our training material will interact on the type of network we'll need.
- Determine how the computer will choose which specific alterations to perform on the melody to make it a reasonable sounding chorale, it can't sound like a machine.
- To prevent it being too boring, we may have to introduce a tonal "wild card", like the ways that Bach pushed against the rules to add flavor and color in his pieces.
- Unsure if taking a web based app or a desktop based app would be appropriate.
- Allowing user input requires the network to train on this input and then on a data set, meaning there would be a large wait time after inputting, unless we can train on the data set and then train on the input separately.

3. Functionality

There are a set of rules that one can implement to harmonize a basic melody to create from it a full chorale, and these alterations are fundamental in the writing of new music of all kinds. Some of these rules include the standardized singing ranges (soprano, alto, tenor, and bass). Another rule is, when given a consonant root-position triad, you may omit the fifth if this results in smoother voice leading while also doubling the bass. You may not omit the third. These rules of chorale writing will be

hard-coded in, taking advantage of the numerical relations between the pitches, and similar relationships in the chorale database.

Before we can implement these rules, we need the neural network to generate the soprano voice. To do this, we will use a database of over 100 Bach's chorales that UCI has made public. The network will have iterated over all these chorales to "learn" the relationships between both notes and chords in a line. Then, if given an input by the user, the neural network will apply its "knowledge" to complete the line. If told to create its own it will have free reign over what it creates. Once this is finished, it will be passed to the chorale rules dictator, which will design the other three voices of the piece based on those rules we discussed.

4. Timeline

Weekends will be used to finish or start sections early, general timeline will follow school weeks.
January 18th (Wed.) to January 27th (Fri.):
Research on individual pieces and planning the design
- How to design and train a neural network on chorales
- Chorale rules and generation from a single line
- Turning our abstract representation of music into sheet music and sound
- Using jMusic library in getting basic file setup ready

January 30th (Mon.) to February 17th (Fri.):
Understanding, coding, and testing the neural network
- Following tutorial on building neural network
- Design it to learn from previous Bach chorales
- Be able to produce single lines of music that follow Bach's form

February 20th (Mon.) to March 3rd (Fri.):
Coding and testing the Chorale Builder
- Takes a soprano music line and builds a chorale around it
- This will use Bach's basic tenets of chorale writing
- At this point, our project will be able to create chorales by itself

March 6th (Mon.) to March 17th (Fri.):
User input line to build Chorales on
- User input that the neural network can read and expand to full length
- Then, that line is fed into the chorale builder and completes the choral
- By this point, project should be fully functioning, ignoring user interface

March 20th (Mon.) to April 14th (Fri.):
Data display and User interface implementation
- Creating a user interface, in app or in browser
- Prompts for a short music line on a music staff or other options from user
- Neural network then does its thing, and generates a chorale
- Finally the user can view, play, and print the resulting chorale, or restart

April 17th (Mon.) to April 28th (Fri.):
Prepare a presentation
- Pretty obvious what this is
- May want to make a tri-fold detailing process as well as a live demonstration

May 1st (Mon.) to end of semester:
Write the paper about or capstone
- Describing the design process we went through, our final product, etc.