MiNee Me: Exploring Music Composition Through Robotics

Nicolas Nee, COS ‘24
Independent Work Advised by Radhika Nagpal and Allison Chen

My Two Goals:

1. Could I create a robot that could play the piano?

2. Would I be able to create a prediction model that could make its own music?

Robot Design Decisions

- I used MIDI files originally made to digitally transfer and edit music.
  - The audio is stored by track and messages, which contain all note information
- Solenoids to push the keys
  - Initially I wanted to use servo motors because they were much less complicated, but they could not move fast enough

MIDI Files

- Loop over the MIDI file prior to passing information to the Arduino
  - For each significant time, save the notes that are interacted with
- Next, loop over the total time and send the list of notes interacted with to the Arduino

Solenoids

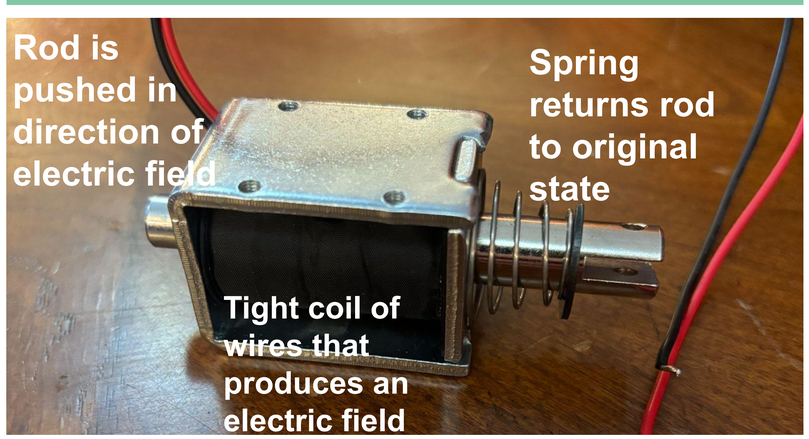

Fig. 2: A diagram showing the solenoids used in this project

Next Note Predictor

- Training
  - To train the model, I kept track of the previous 10 notes for a specific input and added them into a 3D input array and a 2D output array. I then used one hot encoding to keep track of what note came after a certain 10 note sequence.
- Data Cleaning
  - For each Nintendo file that I downloaded, I added each note to another file, train_notes.txt. I ended up making multiple different types of models, such as filtering every note to the 7 white notes in Middle C so that MiNee Me could play.
- Generating
  - To generate, the function takes in a sequence of 10 notes and returns the note that the model believes should appear after it.

```
MidiFile(type=0, ticks_per_beat=384, tracks=[
  MidiTrack([
    MetaMessage('time_signature', numerator=4, denominator=4, clocks_per_
    MetaMessage('set_tempo', tempo=1090909, time=0),
    MetaMessage('track_name', name='Electric Piano', time=0),
    Message('program_change', channel=0, program=0, time=0),
    Message('note_on', channel=0, note=48, velocity=50, time=0),
    Message('note_off', channel=0, note=48, velocity=0, time=96),
    Message('note_on', channel=0, note=50, velocity=50, time=0),
    Message('note_off', channel=0, note=50, velocity=0, time=96),
    Message('note_on', channel=0, note=52, velocity=50, time=0),
    Message('note_off', channel=0, note=52, velocity=0, time=96),
```

Fig. 1: The output of a MIDI file and how I was able to read it.

Model Results

The output is largely constrained by the fact that only one note can be activated or deactivated at a time. The first improvement that I could have made is adding a way to recognize chords. The second would be to gather music from the same key, so that the patterns could be more easily recognizable. As seen in figure 4, the validation accuracy is quite low, largely due to this problem. I did not have enough similar data to solidify patterns even though I had around 860,000 notes to train off of.

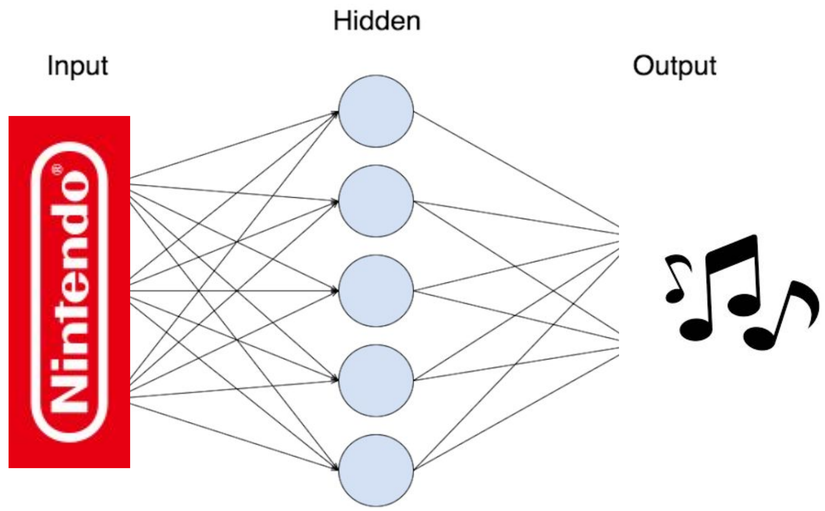

Fig. 3: The model used Pokemon and Mario music to predict new notes

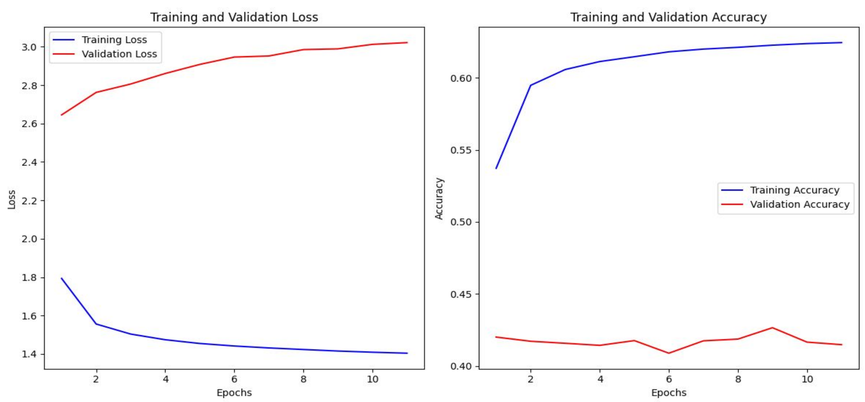

Fig. 4: The training and validation loss and accuracy over different epochs.