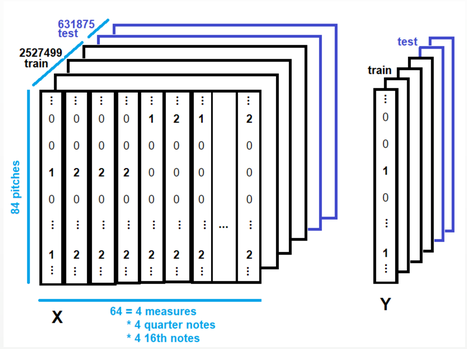
Music Generation With Deep Neural Networks Using Flattened Multi-Channel Skip-3 Softmax and Cross Entropy

Jessica Myers
Senior Thesis, CSSE Department, Rose-Hulman Institute of Technology

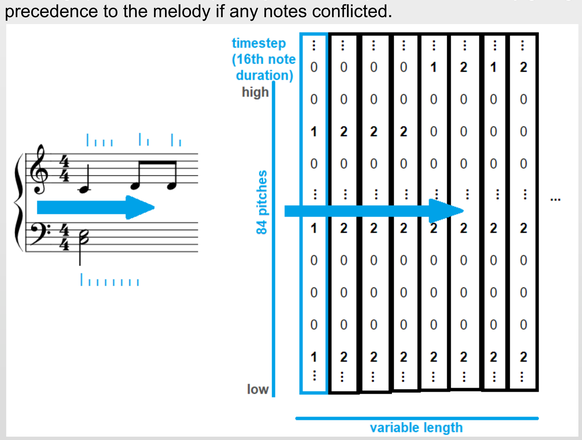
Introduction

Music is governed by rules—the time signature restricts how many notes can fit in a measure, and Western music has chord progression flowcharts—so it only seems natural to code these rules to generate music. But what about creativity? Composers often break rules to be creative. This is where a rules-based perspective breaks down. Non-musicians don't enjoy music based on strict rule-following; yet they somehow know when something is off [1]. This implies a sense of passive learning over the duration of a person's life, which led to an investigation of deep neural networks as a means of generating music.

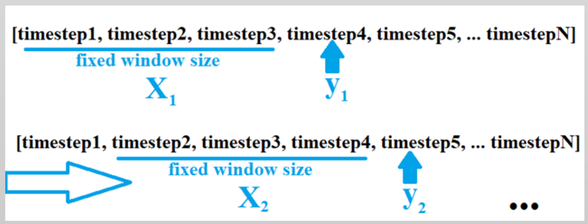


Custom Dataset

The raw dataset of MIDI files was acquired from Hooktheory's TheoryTab Database [2] using a slightly modified version of a web crawler created by Wen-Yi Hsiao on GitHub under the project name "Lead-Sheet-Dataset" [3]. The raw dataset consisted of **17,216 song clips** from **4825 artists**. Only two-track "no-key" (key of C) files were kept. MIT's Music21 Python library [4] was used to iterate over the melody and chord parts of each MIDI file. In this context, a **timestep** is defined as an array of size 84 representing **84 distinct pitches** (7 octaves) with one 16th note duration, consisting of **{0, 1, 2}**, which respectively represent a rest, a pitch begins, and a pitch is held. For each note, chord, or rest object in the MIDI file, timestep(s) were created for that object's duration. The two resulting sparse matrices (melody and chords) were combined by giving precedence to the melody if any notes conflicted.

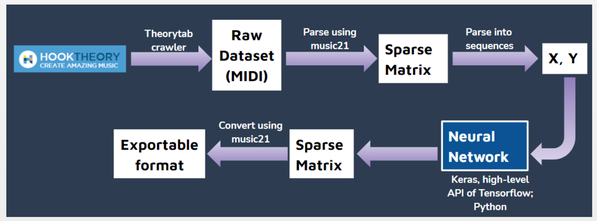


Padding consisting of 4 measures of rest was added to the front of each song. The final dataset was then generated by sliding a fixed window, 4 measures (**64 timesteps**) wide, over each song, generating (X, y) pairs, with the target being the next note after the 4 measures.



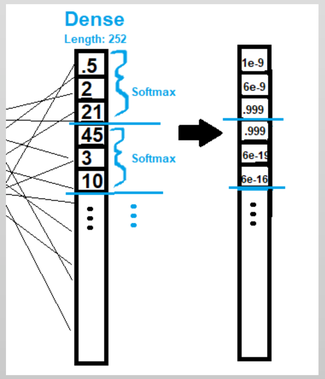
The dataset was split 80%/20% train/test and then shuffled.

System Architecture



Custom Loss and Activation Functions

This seemed to be a clear classification problem (between the values of 0, 1, or 2). However, instead of a 1-dimensional list of possible targets, as would be typical of a classification problem, this problem consisted of a 2-dimensional array of 3 possible categories for each of the 84 possible pitches that could be played at any given time. Visualizing this as 3 channels for each of {0, 1, 2}, similar to a color picture's RGB channels, the goal was to take the softmax across these channels for each of the 84 pitches. However, the output of the network was a fully-connected, flattened layer of 252 (84 * 3) output nodes. Using TensorFlow Keras, the traditional softmax activation function did not work with this setup. Thus, digging into the internals of Keras, a **custom skip-3 softmax function** was created that takes the softmax across every 3 nodes. In addition, a **custom skip-3 cross-entropy loss function** was created that operates over every 3 entries (mimicking 3 channels), as well as a **custom skip-3 accuracy**. As far as the author knows, this is a **novel** approach in the domain of music generation due to this specific matrix representation of music.



Neural Network Architectures

1. **Fully-Connected Network**

```
model = Sequential()
model.add(Input(shape=(64,84,1)))
model.add(Flatten())
model.add(Dense(800,activation='relu'))
model.add(Dense(500,activation='relu'))
model.add(Dense(300,activation='relu'))
model.add(Dense(252,activation=skip3_softmax))
```

2. **Convolutional Network**

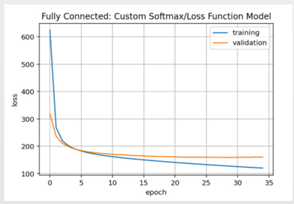
```
model = Sequential()
model.add(Input(shape=(64,84,1)))
model.add(Conv2D(20,kernel_size=(4,18),padding='valid',activation='relu'))
model.add(Flatten())
model.add(Dense(800,activation='relu'))
model.add(Dense(500,activation='relu'))
model.add(Dense(300,activation='relu'))
model.add(Dense(252,activation=skip3_softmax))
```

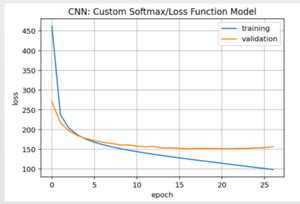
```
model.compile(loss=skip3_cross_entropy,
              optimizer=Adam(learning_rate=0.00001),
              metrics=[skip3_accuracy])
```

Results

To put things into perspective, here is the first sound this project produced (Fully Connected, 1000 data records, no padding in front of the dataset):

Both networks were trained with 500,000 sequences, a process that took several days.





Trial	Seed	CNN Result	Analysis	Fully-Connected Result	Analysis
A	"Twinkle Twinkle Little Star"		Repeats seed's chord progression, but no melody. Last note resolves.		Frequent pauses, more melodic than CNN. Last note resolves.
B	Chords only (front-padded with rests)		Keeps a consistent rhythm (no "melody" just chords)		Keeps a consistent rhythm, but doesn't resolve.
C	Unresolved Scale (stops on the leading tone) (front-padded with rests)		Continues the leading tone, then resolves on the tonic. Silence, then plays the dominant (scale degree 5)		Continues the leading tone, then silence. Doesn't resolve.
	***Note that any silence observed in the last 2 seconds of these mp3s is the result of the mp3 file format, not the actual song clip.				

Conclusion

Strengths: Both networks seem to generate continuing notes that belong to the same chords that the seed had. In other words, it doesn't sound entirely random. All the chosen notes are within the same range as the seed (they are not extremely high or low in comparison). Both networks seem to continue or create rhythms on beat, fairly consistently. There is evidence of rhythmic patterns. There are hints of chord progressions, some continued exactly from the seed, and the networks seem to be good at endings (often resolving on the tonic).

Weaknesses: Both networks seem to want to end the song. Often, the generated music consists of a somewhat resolved ending to the seed, followed by silence that keeps being generated if it is left to keep generating. This can lead to abrupt, near-sighted endings. Sometimes it seems like the noise becomes amplified the longer the model generates a continuation. Both models favor chords over melody. There appears to be no sign of a melody, aside from the chords. As for creativity, the generated music seems to barely grasp musical intuition. Perhaps creativity is an emergent property that comes about when the basics are in place. At this point, the networks do not seem to be able to replicate a melody, which makes it difficult to analyze. The generated music seems to depend largely on the seed given, with results varying widely, and sometimes the generated notes sound "wandering" and aimless.

Future Work

Now that the musical pipeline is fully established, there are some specific areas for improvement. Other architectures should be tested beyond the simpler ones that have been created here. A convolutional network with more convolutional layers could cause more aspects of music theory to be learned. In addition, incorporating attention or dilated convolutions would be a next step to try to make the network less near-sighted so it produces more goal-oriented music. Another improvement could be to reduce the scope from melody and chords to just melody, in order to focus the network's training as well as post-generation analysis. Given more time, the entire dataset can be used in training instead of only 500,000 sequences. Lastly, results of the convolutional network could be explored in more depth by looking at the particular weights of each filter to try to figure out what properties of music theory the network is learning.

References

[1] Bharucha, Jamshed J. and Peter M. Todd. "Modeling the Perception of Tonal Structure with Neural Nets." Computer Music Journal, 1989, vol. 13, no. 4, pp. 44-53.
[2] https://www.hooktheory.com/theorytab/artists/
[3] https://github.com/wayne391/lead-sheet-dataset
[4] http://web.mit.edu/music21/
[5] Yang, Li-Chia, et al. "Midinet: A Convolutional Generative Adversarial Network For Symbolic-Domain Music Generation." ISMIR, 2017.
[6] Chuan, Ching-Hua and Dorien Herremans. "Modeling Temporal Tonal Relations in Polyphonic Music Through Deep Networks with a Novel Image-Based Representation." Thirty-Second AAAI Conference on Artificial Intelligence, 2018.
[7] Oord, Aaron van den, et al. "Wavenet: A Generative Model For Raw Audio." 2016.
[8] Elbayad, Maha, et al. "Pervasive Attention: 2D Convolutional Neural Networks for Sequence-to-Sequence Prediction." 2018.
[9] Carnovalini, Filippo and Antonio Rodà. "Computational Creativity and Music Generation Systems: An Introduction to the State of the Art." Frontiers in Artificial Intelligence, 2020, vol. 3, no. 14.