# Polyphonic Music Generation from MIDI Performances Using RNNs

Sean Metzger (seanmetz) & Daniel Gallegos (dglaort)

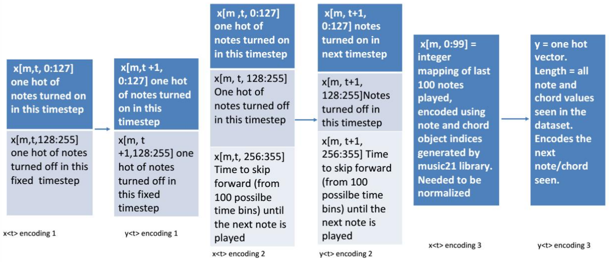
## Project Goal and Motivation

Composing music is one of humanity's most amazing accomplishments. We thus sought to generate high quality classical music using an LSTM. Although many LSTMs can generate decent music, We sought to create an LSTM that could not only support multiple notes at once, but also encode dynamics, making the music sound less robotic and more human generated.

We trained our LSTM on 100-note sequences of classically played music with dynamics generated by musicians on the Yamaha E-Piano Competition Dataset. The dataset contains performance of the greatest classical pieces ever by Bach, Chopin, Beethoven and more.

## Encoding MIDI Files

Each vector x represents a single time step from our data.
Each vector y represents the desired output for one step of the RNN.



x[m,t, 0:127] one hot of notes turned on in this timestep

x[m,t,128:255] one hot of notes turned off in this fixed timestep

x<t> encoding 1

x[m,t +1, 0:127] one hot of notes turned on in this timestep

x[m, t +1,128:255] one hot of notes turned off in this fixed timestep

y<t> encoding 1

x[m ,t, 0:127] one hot of notes turned on in this timestep

x[m, t, 128:255] One hot of notes turned off in this timestep

x[m,t, 256:355] Time to skip forward (from 100 possilbe time bins) until the next note is played

x<t> encoding 2

x[m, t+1, 0:127] notes turned on in next timestep

x[m, t+1, 128:255]Notes turned off in this timestep

x[m, t+1, 256:355] Time to skip forward (from 100 possilbe time bins) until the next note is played

y<t> encoding 2

x[m, 0:99] = integer mapping of last 100 notes played, encoded using note and chord object indices generated by music21 library. Needed to be normalized

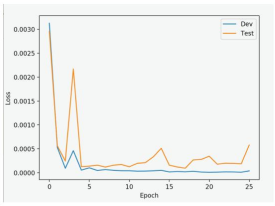
x<t> encoding 3

y = one hot vector. Length = all note and chord values seen in the dataset. Encodes the next note/chord seen.

y<t> encoding 3

### Choosing an Encoding

Despite low loss, our initial results were bad due to our very sparse encodings which made us change encoding schemes.



### Representing Polyphony

Chords are the backbone of music and our model would have been incomplete without them. Our first two models had polyphony encoded as a one hot vector that was on in several locations at once, which made decoding prediction vectors from our LSTM very difficult.

Using the Music21 Library from MIT allowed us to encode sets of multiple notes as a chord object

This allowed us to enumerate notes played at the same time as chords

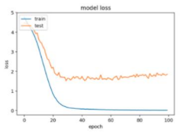
De-emphasizes notes being played at the same time in the end, but is a clean representation.

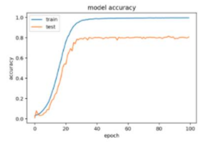
## Architecture Selection & Hyperparameter Tuning

| Initial Model Architecture Search | LSTM Cell (256), 1 Dense Layer | 2 Stacked LSTM Cells (256), 1 Dense Layer | 2 Stacked LSTM Cells (128), 1 Dense Layer | 2 Stacked LSTM Cells (512), 1 Dense Layer | GRU Cell (256), 1 Dense Layer |
|---|---|---|---|---|---|
| Train Accuracy | .994 | .995 | .995 | .995 | .9956 |
| Test Accuracy | .771 | .8065 | .8065 | .8065 | .8141 |

| 'Panda' Model Improvement Highlights – | Dropout (p = 0.3) with 2 Stacked LSTM cells | Aggressive Dropout (p = 0.6) + Regularizing Kernel in LSTM | Stacked GRU Cells with Dropout | 4 Stacked LSTM Cells with 4 Dropout Layers, 1 Dense Layer |
|---|---|---|---|---|
| Training Accuracy | .995 | .986 | .986 | .995 |
| Test Accuracy | .8241 | .8216 | .8216 | .8568 |
| Train Loss | .0402 | .0499 | .0499 | .0135 |
| Test Loss | 1.56 | 1.52 | 1.52 | 1.518 |

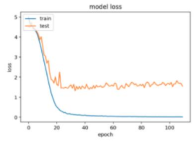
| Hyperparameter choice | Sequence length | Learning Algorithm | Initial Learning Rate | miniBatch Size | Number of Epochs |
|---|---|---|---|---|---|
| Final Value | 100 | RMS prop | .01 | 100 | 60 |

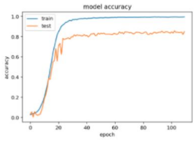




Initial vs. final model loss metrics

Initial vs. final model accuracies



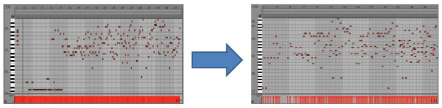


## Results and Future Steps

### Listening to our results & post-processing our model's output

Randomly selected one time sample from our dataset and propagated our RNN forward to generate samples
After getting our generated midi files, we wanted to do some post processing to make it sound better. These are some of the things we tried:

- Delete the first n frames. It took the model to warm up a bit before outputting interesting melodies.
- Remove repeated notes. A lot of times the model repeated the previous note when it did not have anything other note it wanted to play immediately, so we took these as rests.



### Future Work

1. Modifying the encoding so that we can create notes that are held for multiple beats. This can be combined with a little post processing to create notes that can take any length time.
2. Modifying the encoding to create notes that can play more loudly and softly. This combined with the first one can be used to create a program that can generate the full extent of songs able to be played on the piano.
3. We could alter the loss function to take into account harmonics and playing a note that doesn't cause dissonance. This would help the model train better and generate music that more closely mimics piano pieces from the baroque, classical, and romantic era.
4. Create a more cohesive dataset so that it can generate music better of a more specific type instead of generic "classical" music.

## References & Acknowledgements

1. https://magenta.tensorflow.org/
2. https://towardsdatascience.com/how-to-generate-music-using-a-lstm-neural-network-in-keras-68786834d4c5
3. Jazz Improvisation Coursera lecture
4. https://arxiv.org/pdf/1606.04930.pdf
5. https://arxiv.org/pdf/1412.3191.pdf
6. The AWS support guy

CS 230 – Winter 2018 – Stanford

seanmetz@stanford.edu and dgalort@stanford.edu