DJamBot: AI Music Generation

Daniel Dore (ddore@stanford.edu), Joey Zou (zou91@stanford.edu)

Predicting: We worked on the problem of using a neural network to compose music. More specifically, the model is trained on a database of musical "scores" (coming from MIDI files) and learns to predict the notes played at the next time step given the notes playing at the current timestep. Our model incorporates music theory, performance dynamics, and harmony (i.e. multiple notes played simultaneously).

Data: We used MIDI files produced from the Yamaha Piano-E-Competition (http://www.piano-e-competition.com/ecompetition/default.asp). These files are recorded from professional pianists playing classical (and romantic, etc.) music on digital pianos. This dataset has the advantage of containing information about dynamics (recorded in MIDI as "velocities") based on how hard performers hit the keys on the digital piano recording their performances. It is also somewhat stylistically homogeneous, in that it consists solely of solo instrumental piano music.

We performed data processing to transform the MIDI files into "pianoroll" files which turns the file into an rank-3 array where the (t,n,i) entry indicates whether note n was played at time t (for i=0) and, if played, how loudly it was played (i=1). During data processing, we also normalize the keys and tempos to be mostly uniform across the dataset, and determine the chord progression.

Features: We started with the implementation in JamBot, which approaches music generation as two separate learning problems. One neural network learns to predict the next chord in a chord progression. A second neural network learns to predict the notes at the next timestep based on the current chord and currently playing notes.

We added the feature of predicting note "velocities" (i.e. dynamics) as well. More specifically, the network learns two vectors at each timestep: the first is a vector of probabilities which specifies which notes are to be played, and the second is a vector of velocities which specifies how loud to play each note.

Models: We followed the LSTM approach of JamBot. Since we want to predict both the notes played and the velocities of the played notes, we used a custom loss function. We treated learning which notes to play as a series of binary classification problems: is note n on or off? Thus, we used binary cross-entropy as the loss function for the probability vector. For the velocities, this loss function is inappropriate, so we used mean-squared error. However, using just mean-squared error results in poor performance, as it over-penalizes the model for predicting non-zero velocities for notes which are "off". Thus, following the approach in the DeepJ paper, we only include terms in the mean-squared error computation corresponding to notes which are on:

$$\mathcal{L}(n_{\text{true}}, n_{\text{pred}}, v_{\text{true}}, v_{\text{pred}}) = \frac{1}{N}\left[\sum_{i=1}^{N}(n_{\text{true}}^{(i)}\log(n_{\text{pred}}^{(i)}) + (1-n_{\text{true}}^{(i)})\log(1-n_{\text{pred}}^{(i)})) + \sum_{i=1}^{N} n_{\text{true}}^{(i)}(v_{\text{true}}^{(i)} - v_{\text{pred}}^{(i)})^2\right].$$

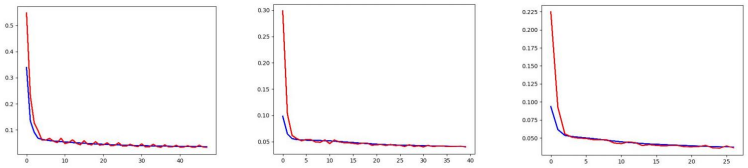
One inherent difficulty in the problem of music generation is that there is no good performance metric apart from the loss function itself. Thus, we must subjectively evaluate our model's performance and decide whether it sounds "like music".

Discussion: Despite experimenting with a variety of architectures, input representations/normalizations, and parameters, we have been unable to produce a model with all of our desired features that produces meaningful music. Several of our models have produced very low probabilities of playing any notes whatsoever, very low velocities, or music that sounds like a single note being played. The problem seems to be not with the neural network itself, but with the data processing: our loss function decreases nicely.

Future: We would rewrite the code from scratch and try to isolate the cause of the poor performance. We could also experiment with a larger variety of neural network architectures, such as varying the amount of regularization, dropout, etc. It would also be interesting to add more features, such as adding an attention mechanism to allow more long-term structure.

Results:

Model Features	Training set size	Test set size	Training error	Test error
1 LSTM	324	36	0.0323	0.0362
2 LSTM	324	36	0.0394	0.0408
2 LSTM + Bidirectional	324	36	0.0361	0.0374



References:

- Brunner, Gino, et al. "JamBot: Music Theory Aware Chord Based Generation of Polyphonic Music with LSTMs." *arXiv preprint arXiv:1711.07682* (2017).
- Mao, Huanru Henry, Taylor Shin, and Garrison W. Cottrell. "DeepJ: Style-Specific Music Generation." *arXiv preprint arXiv:1801.00887* (2018).