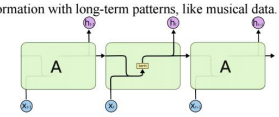
Music Generation Using Recurrent Neural Network Models

Gabriel Voorhis-Allen, Cade May, KK Mokobi
Department of Computer Science, Stanford University

Introduction

For this project, we sought to produce a generative model which could produce artistic results after being trained on human-made creative art. Music is a very artistically rich domain, but can also be broken down into data (as it simply consists of sequences of pitch frequencies at varying rhythms and tempos). Thus, we decided to create a model to perform classical musical generation We chose to train a recurrent neural network (RNN) with Long-Short Term Memory (LSTM), a type of neural network used for sequential information with long-term patterns, like musical data.



3 Layer LSTM (source:google.com)

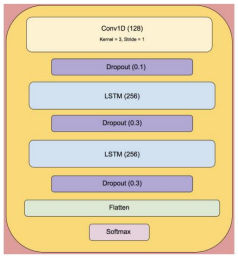
Data

- Music stored in MIDI musical files (MIDI files effectively package musical data and allow for its manipulation)
- 10/3/3 dataset split; 10 songs for training set, 3 for development, and 3 for testing.
- The training set had 11,000 sequences of musical data of length 100ms each.
- Musical genre was piano-only classical music, which allowed the model to focus on producing the best single-track output possible, instead of being required to optimize multiple-track and multiple-instrument data.
- Datasets consists only of songs with a 4/4 time signature, allowing the model to focus on rhythm and pitch sequencing without encountering conflicting beats per measure.

Model

We evaluated a wide range of architectures. Most of the models we worked with involved a recurrent neural network base. On top of this GRU or LSTM base, we experimented with Convolution1D, TimeDistributed (Dense), BatchNorm, Separable Convolution, and more.

All of the training and testing accuracy values displayed in the results section were produced by models that were trained for 200 epochs, except for the 2nd model, labeled "Conv1D, 2 LSTM-256, Dropout." This model was trained for 860 epochs. Based on promising results that we found in the outputs of this model in its early stages, we decided to train it for a lot longer. It is from this model that we obtained our best music sample. This is the model that is displayed in the figure to the left.



Data Processing

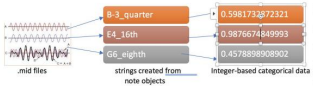
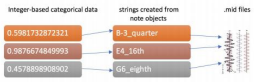


Fig 1: MIDI files converted to notes using music21 parsing function, and a mapping function is used to create integer-based training data

Fig 2, right: Predicted results converted to notes using a mapping function, and written to MIDI files using music21 functions



1) Convert data from .mid files to integer-based categorical data using mapping functions and functions from music21 package
2) Fit model: calculate loss using categorical cross entropy
3) Generate music using model to predict integer values that we then convert to notes, and use music 21 stream to write to MIDI file

Results, Hyperparameter Tuning, and Architecture Search

Build:	Baseline: 4 LSTM-256, Dropout	Conv1D, 2 LSTM-256, Dropout	1 LSTM-256, Dropout	Conv1D-128, 2 GRU-256, Dropout	SeparableConv1D, 2 LSTM-256, Dropout	Conv1D-256, 3 LSTM 128, Dropout	Conv1D, 2 LSTM-256, BatchNorm
Train Acc:	0.94	0.93	0.94	0.95	0.94	0.94	0.93
Test Acc:	0.60	0.88	0.68	0.79	0.94	0.72	0.73

Table 1 above displays the qualitative results of some of our models.

Discussion of Results

- These results were the result of evaluation of the generated music, both in terms of empirical study (cross-entropy loss) and qualitative analysis (conducting surveys on music quality of various models).
- We spent a lot of time on architecture search, exploring a wide breadth of models, but the superior performance of the Conv1D/LSTM-256 model, which was trained over 860 epochs, indicates that many of the other models could have benefitted from further epochs of training.

Discussion of Results (contd.)

- With respect to our quantitative metrics, all of our models exhibited characteristics of high variance, as shown in Fig 1.
- The numerical evaluation of our models involved the categorical cross-entropy function.

Fig: An sample of our model's MIDI output, displayed in Garageband

$$CCE = -\frac{1}{N}\sum_{i=0}^{N}\sum_{j=0}^{J} y_j \cdot log(\hat{y}_j) + (1 - y_j) \cdot log(1 - \hat{y}_j)$$

The categorical cross-entropy loss function.

Summary and Future Work

We had lofty goals to start this project: we envisioned a general-purpose music generator which, given a MIDI file of music, would produce similar-sounding music. We were able to accomplish our core task; the different models we trained generate music of varying degrees of quality, and our best model (a two-layer LSTM network with a Conv1D layer and dropout) produced adequate-sounding music after being trained for 860 epochs. However, our project could be improved by loosening the simplifying restrictions that we put in place along the way.

We plan to improve our networks to be robust to larger and more complex datasets: music from a wider array of genres, with more instruments, and with varied time signatures. We believe that the model we've trained will be largely transferrable to accomplishing this more complex task, although we will need to perform many more iterations than the 860 we achieved this time. We do plan further changes to the model itself as well, including incorporating dimensionality reduction to produce less complex, but better-sounding output.

References

- D. Gallegos and S. Metzger. "LSTiestoM: Generating Classical Music." CS230: Deep Learning, Winter 2018.
- Skúli, Sigurður. "How to Generate Music Using a LSTM Neural Network in Keras." Towards Data Science, Medium, 7 Dec. 2017.