Jul 27th, 3:15 PM - 4:15 PM

Using Long-Short Term Memory Network to Train Machine Composing Baroque Fugue/Canon

Yihe Chen
Susquehanna University

Toshiro Kubota
Susquehanna University

Using Long-short Term Memory Network to train machine composing Baroque Fugue/Canon

Author: Yihe Chen Advisor: Toshiro Kubota Ph.D
Department of Mathematical Sciences

WHAT IS FUGUE/CANON, WHY THEM?

- **Fugue**: is a contrapuntal compositional technique in two or more voices, built on a subject (a musical theme) that is introduced at the beginning in imitation and which recurs frequently in the course of the composition. A typical fugue usually have three sections: an Exposition, a Development, and a final entry that usually contains the return of the subject in the fugue's tonic key. Here in this case, a tiny variant of typical fugue, which is Ricercar, will be more preferred, as a type of late Renaissance and mostly Baroque instrumental composition.

- **Canon:** a contrapuntal compositional technique or texture that employs melody with one or more imitations of the melody played after a given duration

Canon and Fugue as polyphonic musical texture were not practically separated until early1600s.

 # A BRRIEF ILLUSTRATION TO THE STRUCTURE OF A CANON:

PART I: ABCDEFG ABCDEFG ABCDEFG...
PART II: ABCDEFG ABCDEFG ABCDE...
PART III: ABCDEFG ABCDEFG ABC...

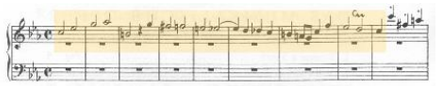
Duration on delay entry on Part ii

 # A BRRIEF ILLUSTRATION TO THE STRUCTURE OF A FUGUE:

PART I: A - ~~~ A'~~ ~~ ~A'~ ~~ ~~~
PART II: A' ~~ ~A'~ ~~ A'~~ ~~ ... ~~~
PART III: A' - ~A'~ ~~ ~A'~ ~~ ~~~

Entry Episode Entry Episode Coda

* A: *Subject* A': *Answer to the Subject* ~: *free conterpoint*



"Regis issu cantio et reliqua canonica arte resoluta"
"The theme given by king, with addition, resolved in canonic style"

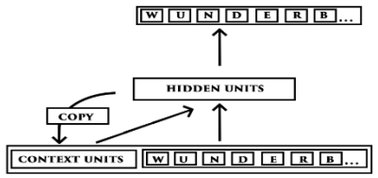
Known as *die Thema Regium(the king's theme)*, it is the theme that was given from King Friedrick II of Prussia to Johan Sebastian Bach as the subject to his set of Fugues and Canons which was later to be known as *das musikalisches Opfer*, the Music Offering.
Figure i & ii

Within such musical textures under relatively strict rules and laws to follow, it is our believe that those rule could have made composing Fugue/Canon–like music a relatively easier work to be taught to machinery.
In this project, the training data are primarily keyboard Fugue/Canon pieces composed by Johan Sebastian Bach.

WHAT IS A LONG-SHORT TERM MEMORY NETWORK(LSTMs)?

An LSTMs is an implementation of Recurrent Neural Network (RNN) using Long-short term memory architecture.

RNN:
Unlikely normal feed-forward networks, a RNN network will not only take its current input example they see, but also what it have perceived one step back in time. The decision a RNN have reached at time step *t-1* affects the decision at time step *t*. Thus, the network has two sources of input: the present and the recent past, which will be combined to determine how the response to new data would be.



$$\mathbf{h}_t = \phi(W\mathbf{x}_t + U\mathbf{h}_{t-1}),$$

Figure iii:
A brief illustration to RNN's feedback loop, and the mathematical expression of carrying memory forward. In the expression, h_t represents hidden state at time step t, x_t represent the *input at time* t, modified by a weight matrix W added to the hidden state of the *previous time step* h_{t-1} multiplied by its *hidden-state-to-hidden-state matrix* U, otherwise known as a transition matrix and similar to a Markov chain.

LSTMs:

However, in a traditional recurrent neural network, during the gradient back propagation phase, the gradient signal can end up being multiplied a large number of times by the weight matrix associated with the connection between the neurons of the recurrent hidden layer. When the weights in this matrix are small, it can lead to a situation called *vanishing gradients* where the gradient signal gets too small and the learning becomes very slow. It also makes learning long-term dependencies difficult. If the weights in the matrix is overly large, it can lead to a situation called *exploding gradients* where the learning process diverges.

The LSTM model was designed to compensate the above issues. It introduces a new structure called a *memory cell*. A memory cell is composed of four elements: an input gate, a neuron with self-recurrent connection, a forget gate, and a output gate.

Figure iv: Illustration of an LSTM memory cell

The equations (1)-(6) show how a layer of memory cells is updated at every time step *t:*

* x_t as the input to memory cell layer at time t.
* W_i, W_f, W_c, W_o, U_i, U_f, U_c, U_o, *and* V_o as weight matrices
* b_i, b_f, b_c and b_o as bias vectors.

We first attempt to compute the input gate value i_t,and $\tilde{C}_t$, the candidate value for state of the cell at time *t*:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \quad (1)$$
$$\tilde{C}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \quad (2)$$

Then compute the value of f_t, the activation of the memory cells' forget gate at time *t*:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \quad (3)$$

With the values gained above, we now can compute C_t, the cells' new state at time *t*:

$$C_t = i_t \times \tilde{C}_t + f_t \times C_{t-1} \quad (4)$$

With the new state of the memory cells, now we can compute the value of output gates, and subsequently the outputs.

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + V_o C_t + b_o) \quad (5)$$
$$h_t = o_t \times \tanh(C_t) \quad (6)$$

THE APPLICATION

Training data pre-processing:

Like a linguistic expression, a piece of music is a sequence of information, and its basic morpheme is highly context-dependent.
We adopted a 2-layer Recurrent Neural Network with LSTM designed for training/sampling a character-level language model.
Music data in MIDI will first be converted into text files; we use ASCII expression to represent the sound track, note duration, frequency, intensity and time, which were being considered as the fundamental morphemes (Figure vi).
Then as a further step of data pre-processing, structural key words were converted into single character code-words. (Figure vii).

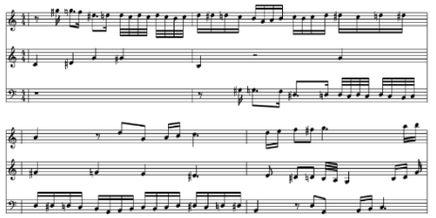


Figure v: a part of BWV 1079 – The music offering, *Canon perpetuus super thema regium*, converted from midi file collected from Dave's J.S Bach Page

```
2, 0, Start_track                    2, 1800, Note_on_c, 0, 72, 0
2, 0, MIDI_port, 0                   2, 1800, Note_on_c, 0, 71, 100
2, 0, Title_t, "String Trio"         2, 1860, Note_on_c, 0, 72, 100
2, 0, Program_c, 0, 48               2, 1860, Note_on_c, 0, 72, 0
2, 0, Control_c, 0, 7, 100           2, 1920, Note_on_c, 0, 71, 0
2, 0, Control_c, 0, 10, 96           2, 1920, Note_on_c, 0, 74, 100
2, 360, Note_on_c, 0, 80, 100        2, 2040, Note_on_c, 0, 74, 0
2, 480, Note_on_c, 0, 80, 0          2, 2040, Note_on_c, 0, 75, 100
2, 480, Note_on_c, 0, 79, 100        2, 2160, Note_on_c, 0, 75, 0
2, 840, Note_on_c, 0, 79, 0          2, 2160, Note_on_c, 0, 74, 100
2, 840, Note_on_c, 0, 77, 100        2, 2280, Note_on_c, 0, 74, 0
2, 960, Note_on_c, 0, 77, 0          2, 2280, Note_on_c, 0, 72, 100
2, 960, Note_on_c, 0, 75, 100        2, 2400, Note_on_c, 0, 72, 0
2, 1320, Note_on_c, 0, 75, 0         2, 2400, Note_on_c, 0, 71, 100
2, 1320, Note_on_c, 0, 74, 100       2, 2520, Note_on_c, 0, 71, 0
2, 1440, Note_on_c, 0, 74, 0         2, 2520, Note_on_c, 0, 67, 100
2, 1440, Note_on_c, 0, 74, 100       2, 2640, Note_on_c, 0, 67, 0
2, 1500, Note_on_c, 0, 74, 0         2, 2640, Note_on_c, 0, 69, 100
2, 1500, Note_on_c, 0, 72, 100       2, 2760, Note_on_c, 0, 69, 0
2, 1560, Note_on_c, 0, 72, 0         2, 2760, Note_on_c, 0, 71, 100
2, 1560, Note_on_c, 0, 74, 100       2, 2880, Note_on_c, 0, 71, 0
2, 1620, Note_on_c, 0, 74, 0         2, 2880, Note_on_c, 0, 72, 100
2, 1620, Note_on_c, 0, 72, 100       2, 3000, Note_on_c, 0, 72, 0
2, 1680, Note_on_c, 0, 72, 0         2, 3000, Note_on_c, 0, 71, 100
2, 1680, Note_on_c, 0, 74, 100       2, 3120, Note_on_c, 0, 71, 0
2, 1740, Note_on_c, 0, 74, 0         2, 3120, Note_on_c, 0, 72, 100
2, 1740, Note_on_c, 0, 72, 100       2, 3240, Note_on_c, 0, 72, 0
```

Figure vi: part of the text file converted from a MIDI file. The column represents: sound track(in this case, the first voice), eventtime, event, time, frequency and intensity

```
2, 0,s                               2, 1800,o, 0, 72, 0
2, 0, MIDI_port, 0                   2, 1800,o, 0, 71, 100
2, 0, Title_t, "String Trio"         2, 1860,o, 0, 71, 0
2, 0, Program_c, 0, 48               2, 1860,o, 0, 72, 100
2, 0, Control_c, 0, 7, 100           2, 1920,o, 0, 72, 0
2, 0, Control_c, 0, 10, 96           2, 1920,o, 0, 74, 100
2, 360,o, 0, 80, 100                 2, 2040,o, 0, 74, 0
2, 480,o, 0, 80, 0                   2, 2040,o, 0, 75, 100
2, 480,o, 0, 79, 100                 2, 2160,o, 0, 75, 0
2, 840,o, 0, 79, 0                   2, 2160,o, 0, 74, 100
2, 840,o, 0, 77, 100                 2, 2280,o, 0, 74, 0
2, 960,o, 0, 77, 0                   2, 2280,o, 0, 72, 100
2, 960,o, 0, 75, 100                 2, 2400,o, 0, 72, 0
2, 1320,o, 0, 75, 0                  2, 2400,o, 0, 71, 100
2, 1320,o, 0, 74, 100                2, 2520,o, 0, 71, 0
2, 1440,o, 0, 74, 0                  2, 2520,o, 0, 67, 100
2, 1440,o, 0, 74, 100                2, 2640,o, 0, 67, 0
2, 1500,o, 0, 74, 0                  2, 2640,o, 0, 69, 100
2, 1500,o, 0, 72, 100                2, 2760,o, 0, 69, 0
2, 1560,o, 0, 72, 0                  2, 2760,o, 0, 71, 100
2, 1560,o, 0, 74, 100                2, 2880,o, 0, 71, 0
2, 1620,o, 0, 74, 0                  2, 2880,o, 0, 72, 100
2, 1620,o, 0, 72, 100                2, 3000,o, 0, 72, 0
2, 1680,o, 0, 72, 0                  2, 3000,o, 0, 71, 100
2, 1680,o, 0, 74, 100                2, 3120,o, 0, 71, 0
2, 1740,o, 0, 74, 0                  2, 3120,o, 0, 72, 100
2, 1740,o, 0, 72, 100                2, 3240,o, 0, 72, 0
```

Figure vii: further processed file: structural key words were replaced by single letters

Training:

Based on Torch framework, the model takes one text file as input and trains a Recurrent Neural Network (Karpathy) that learns to predict the next character in a sequence. The RNN can then be used to generate a sequence of characters that looks like the original training data.
Totally 177 pieces of files were converted and pre-processed, resulted in approximately 143,366 words.

Training setting:

Size of LSTM internal state:	128
Number of layers:	2
Learning rate:	0.002
Learning decay:	0.97
Decay rate:	0.95

By using CUDA, the learning took approximately 16 hours with the following hardware:

CPU:	Intel Xeon E5-2650 v3 2.3 GHz 10 Cores (20 threads)
GPU:	Nvidia GTX 1080 Ti
RAM:	64GB DDR 4 (16 GB x3)
HDD0:	3TB
HDD1:	3TB
SSD:	500 GB

Operating System:	Ubuntu 16.04 LTS 64bit
Torch Version:	v5.2
Cuda Version:	v8.0

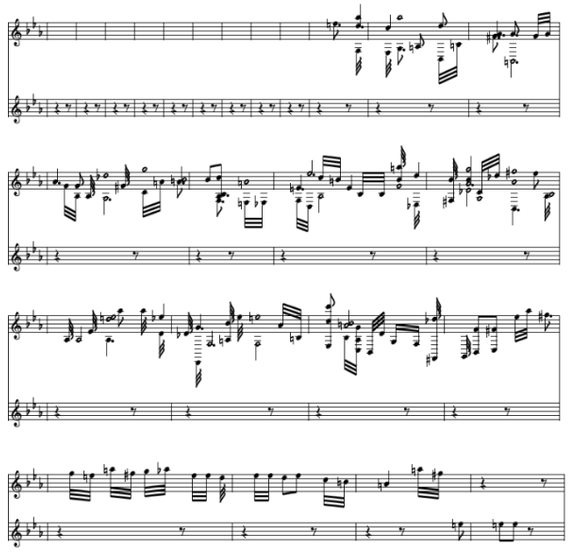
Checkpoints:

While the model is training, it will periodically write checkpoint files for every 1000 iterations (The frequency with which these checkpoints are written could be controlled with number of iteration settings) The filename of these checkpoints contains a very important number: the loss. For example, a checkpoint with filename lm_lstm_epoch0.95_2.0681.t7 indicates that at this point the model was on epoch 0.95 (i.e. it has almost done one full pass over the training data), and the loss on validation data was 2.0681.

Sampling:

The model checkpoint file with lowest validation loss was used to generate asci codes from random seed.
An anomaly filter was programmed and used to correct invalid sequences such as incorrect time-line orders. Then the inverse of the pre-processing procedure was applied to generate a MIDI file.

EXAMPLE RESULT



SOURCES & REFERENCES

- Karpathy. "char-rnn." , https://github.com/karpathy/char-rnn.
- Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780
- Graves, Alex. Generating Sequences With Recurrent Neural Networks. Springer, 2012.

THIS WORK WAS SUPPORTED BY NSF GRANT CCF-1421734.