DeepPiano: A Deep Learning Approach to Music Translation

Hemmatifar, Ali {ahemmati@stanford.edu} and Krishna, Ashish {aak9459@stanford.edu} | CS230: Winter 2018

STANFORD

Motivation

Music notations like ♪ ,♩ , or ♫ are common in daily life. However, playing these notes or assigning meaning happens to be as complex as language processing itself. We aim to use **Deep Learning** to increase understanding of musical language and to translate them into a format suited for assigning location of keys on piano. Every musical symbol provides 2 vectors of information:

- **Duration**: Type of note (filled oval, filled oval with a stem, etc) denotes how long the note is held
- **Pitch**: Audible frequency of the note determined by its location of the staff (horizontal lines with spaces in between)

This work is useful for music instructors and learners alike who may want to play music without needing to learn formal music language. Recent attempts in Optical Music Recognition (OMR) [2] have been encouraging in identifying notations of music. *Our project attempts to expand on recent OMR strategies presented by Jorge, et.al.[1] using Lilypond engraver [3]).*
In our project, we take input in graphical PNG format which represents a sequence of musical notes and output a series of integer labels that can be post processed into XML, JSON, or other custom formats. For interoperability, we have developed a canonical vocabulary.

Results and Discussion

Figure 2. shows results of our training set:

- Notes accuracy defined as total percentage of musical notes correctly labeled in evaluation data set
- Sample accuracy defined as percentage of examples correctly labeled
- Notes and sample accuracy significantly increase from 20k to 40k training data.
- Notes and sample accuracy fixed from 40k to 70k training data.

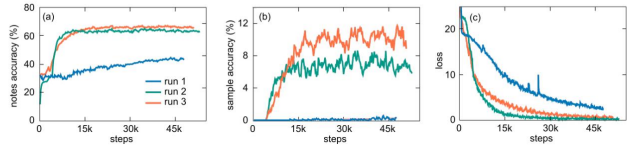

Figure 1: (a) notes accuracy, (b) sample accuracy, and (c) training loss vs steps during training phase.

Dataset Preparation

We used public domain data such as one from [ref Jorge], which, was insufficient so, we generated our own random music sequences. Figure 2. shows sample input sequence with labels corresponding to the symbols. The figure also shows training sample distribution.

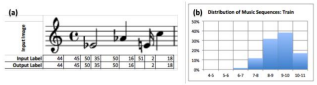

Figure 2: (a) Sample Input; (b) Dataset Distribution

Run Num	Train Size	Test Size	Learning Rate	Num of Steps	Epochs
Run 1	21,000	300	0.01	420	112
Run 2	38,500	300	0.001	770	60
Run 3	73,700	300	0.001	1474	32

Table 1: Dataset Information

DL Architecture

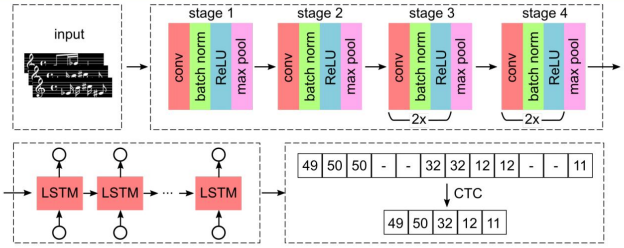

Figure 3: CNN-LSTM-CTC Architecture

Figure 3 represents high level architecture of our DL network which is based on Recurrent Convolution and LSTM, further using Connectionist Temporal Classification (CTC) loss method to collapse sequences and predict blanks (**CNN-LSTM-CTC**). Pre-processed (encoded) data are used for training our RCNN model. We use Adam Optimizer for gradient descent.

Conclusion

1. Our model seems to have threshold for minimum dataset needed for training
2. LSTM performs better with more samples (Run 1 > Run 2 > Run 3)
3. CNN reaches a finite limit beyond 40,000 samples (no improvement in recognizing notes better)
4. In order for better LSTM performance, more randomization of sequences may be needed

References

[1] Jorge Calvo-Zaragoza, Jose J. Valero-Mas, and Antonio Pertusa. End-to-end optical music recognition using neural networks. In *ISMIR*, 2017.

[2] Ana Rebelo, G. Capela, and Jaime S. Cardoso. Optical recognition of music symbols. *International Journal on Document Analysis and Recognition (IJDAR)*, 13:19–31, 2009.

[3] Lilypond. Lilypond music engraving.

Future Work

Rather than random data generation may be used to enhance accuracy. For example, learning in context of musical scales, for example C-Major *(C-D-E-F-G-A-B-C)*. Following rules of musical notes sequencing in scales may introduce grammar that can help improve learning. Also, using PyTorch may increase performance. These could be possible extensions to this project