A PIANO BOY: AN OPTIMIZED MUSIC GENERATOR

Wei Liu, Yinzhi Xi, Jie He

Abstract

Curious about the creativity of neural networks, we developed The Piano Boy, a deep neural network application which can generate piano music by learning existing pieces of piano music.

We implement a 3-LSTM-layers model with an optimized function on rhythms, which has gained the best performance among all comparing models and solved the problem of ending up repeating one note to some extent.

Challenges & Improvements

The goal is to explore a way to improve the existing AI music composers by using different deep neural network architectures or adding creative elements to models. Since this is a generative model, evaluation can rely on the feedback of listeners.

Problem Statements:
1. Could we create a music generator that can develop a sequence of piano music that is quite "realistic", which means very similar to those written by composers?
2. Generally, most models focus on solving the same problem have the problems of rhythms, could we do better?

- Improvements:

The initial implementation cannot output a rest note, we incorporate rest note by reading the midi file every 0.5 seconds, if there is no note or chord at that moment, we add an empty note.
Our next improvement is focused on note/chord representation, in the last two implementations, the note/chord is represented as a word (String), instead, we use an 88 length vector to represent them since piano has 88 keys.
The final improvement is to support multiple keys prediction. In order to generate multiple 1s in the output vector, we modify the activation function from softmax to sigmoid and use a threshold 0.5 to determine if a key is predicted as pressed or not.

Dataset

midi_songs: 92 pieces
Chopin: 98 pieces
Our model uses previous 100 sequence as the input and try to predict the 101st sequence. We use train-test-split for both data sets. For midi_songs dataset, we have 46310 items in training set, for Chopin dataset, we have 58286 items in training set.

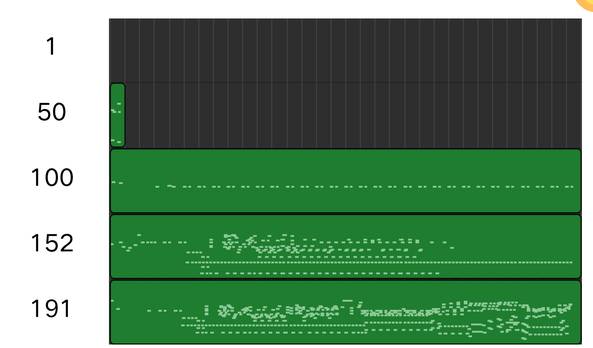
Description of Model(s)

We started from a 3-layer stacked LSTM model and made some modifications to it as comparison. We tested the following variations:
Three LSTM Layers with different dimensionality:
Dimensions in 3 LSTM: (512->512->256)

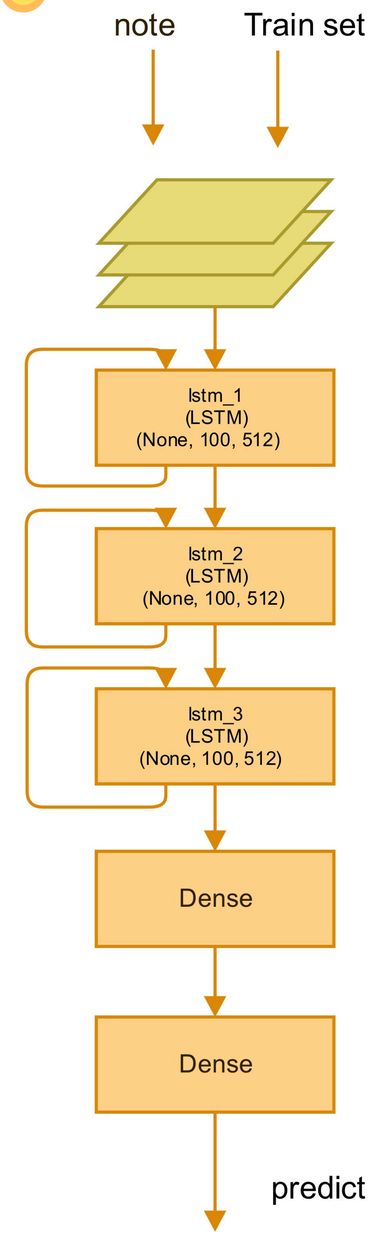
Four LSTM Layers:
Dimensions in 3 LSTM: (512->512->512->512)

Three LSTM Layers with improvements:
Dimensions in 3 LSTM: (512->512->512)
88 keys model 1: predicts multiple possible keys at one beat, it use sigmoid as activation function of output layer and binary cross entropy as loss function, we also use a threshold 0.5 to determine if a key is pressed or not.
88 keys model 2: only predict the key with the most probability, it use softmax as activation function of output layer and categorical cross entropy as loss function.

Different Training Epochs



Model



Related Work

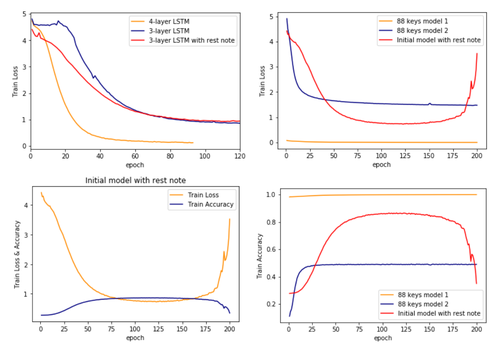
Similar applications:
AI Jukebox: One Bidirectional LSTM
Classical Piano Composer: Three LSTM Layers
Deepjazz: Two LSTM Layers
Deep Jammer: Two LSTM Layers

Results & Analysis

We compared our generated midi file with the outputs of other music generators which are treated as baselines. We conducted a survey to ask listeners to pick the best music product from these outputs. Our model output is a solid leader in the voting results.

Program Name	Number of Votes
AI Jukebox	7
Classical Piano Composer	7
Deep jazz	5
Deep Jammer	4
The Piano Boy	18

We plotted the loss and accuracy for models we trained to observe and evaluate the relationship between cross-entropy loss and epoch over time.



Work Cited

McMahon, B. Music generator utilizing a Bidirectional LSTM architecture in Keras. https://github.com/cipher813/AI_Jukebox
Sigurgeirsson, S. Classical Piano Composer, https://github.com/Skuldur/Classical-Piano-Composer
Svegliato, J. Deep Jammer, https://github.com/justinsvegliato/deep-jammer
Kim, J. Deep learning driven jazz generation using Keras & Theano!, https://github.com/jisungk/deepjazz