Music Generator with Deep Neural Networks

Sina Ehsani

May 2, 2020

1 Abstract / Introduction

With the recent development of machine learning in language models, especially in the deep neural network era, we have seen massive improvement in popular Natural Language Processing (NLP) tasks. Tasks such as machine translation, question answering, and sentiment analysis have been grown exponentially. These advances show the power and potential of these deep learning models.

On the other hand, the advances of NLP are an inspiration to use the methods used in other fields as well. In particular music generation.

Musicians are always seeking out the next best thing, the next best piece of technology which they can utilize to create more compelling music. Specifically, they are always looking for new creative tools that utilize state-of-the-art technology. In addition, writing new, compelling music can be a difficult and daunting task, especially when one is stuck on an idea or lacks new ideas. Anything which can assist in this endeavor is potentially quite useful.

This product, suggests musical compositions to both kickstarts a new music project, as well as provide suggestions for continuations of imported works. It uses deep neural network technology to generate compelling MIDI arrangements.

The MIDI, Music Instrument Digital Interface, which is a communication protocol for digital musical instruments, can be fed to the music generator app as input files, and the application will generate the next lines of music. A deep neural network was applied to our input music sequences to generate the next possible music sequence. Finally, the results are published as MIDI format files.

In one sentence, This product is a music composition tool developed by deep neural networks to automatically generate, extend, and stylize imported music compositions in MIDI.

2 Related Work

- MusNet, by Payne, [5], is a deep neural network project that can generate 4-minute musical compositions with 10 different instruments. MuseNet uses the same general-purpose unsupervised technology as GPT-2 [6], a large-scale transformer model trained to predict the next token in a sequence, whether audio or text.
- Haung et al's Music Transformer paper [3] is an attention-based neural network model that can generate music with improved long-term coherence. Which makes it very relevant to our work.
- The work is done by Simon et al. [9] is

an LSTM-based recurrent neural network designed to model polyphonic music with expressive timing and dynamics. This is an older model that was before the attention Tsunami. It is a good simpler work that is helpful for this project. These two papers were mostly trained with the MIDI piano database.

- Boulanger et al. [1], also applies language modeling to polyphonic music generation. They use an LSTM combined with a NADE. This is also a good related work that can be used to compare results and evaluate our model too.

- Finally, Hawthorne et al. [2] is an automatic polyphonic piano music transcription. With this model, solo piano performances can be converted into MIDI. This might be helpful if we cannot find enough data.

3 Model

For processing MIDI files, the Music21 library was used. Which transfer MIDI files into Music21 representations. Then we picked the highest pick pitch at each semiquaver

After trying several neural network models (including CNN, RNN, Bidirectional RNN, LSTM, and GRU with different sizes and layers), the best performance was achieved using a Bidirectional GRU structure. This model was trained on collected data to make recommendations on user-uploaded midi. The GRU model uses an encoder-decoder model trained to maximize the language modeling objective, i.e. to maximize the total probability of the training sequences.

Figure 1 shows the structure of the deep learning model. In the encoder side, a fixed size of 30 notes has been fed to the network as an input. Our embedding size is 129 (size of vocabulary), which includes:

- 0-127 - note on at the specified pitch
- 128 - note off
- 129 - no event

It is a two-layer GRU model, that each GRU layer has 64 hidden units. The 1st GRU layer is a bidirectional GRU, and the 2nd layer has a single output which is fed into a feed-forward network with a softmax activation function of the size of our vocabulary. Using sparse categorical cross-entropy for loss function and ADAM as the optimizer.

On the decoder side, we have a similar structure to the encoding, the only difference is after each GRU there is a feed-forward network with a softmax activation function to generate the next note. The generated note is fed again to the network as a new input to produce the next note, and this continues until we generate a fixed length of predicted notes.

For regularization of the model we have used dropout and early stopping.

4 Results

For training our model we used the Classical Music MIDI dataset [7]. This dataset consists of classical piano midi files containing compositions of 19 famous composers. And we trained the model for 30 epochs on the dataset. We split this data into training/test sets to evaluate our model. For testing our model, we used five artists from the MIDIworld website [8]. The artist was chosen from the ones not included in the training dataset.

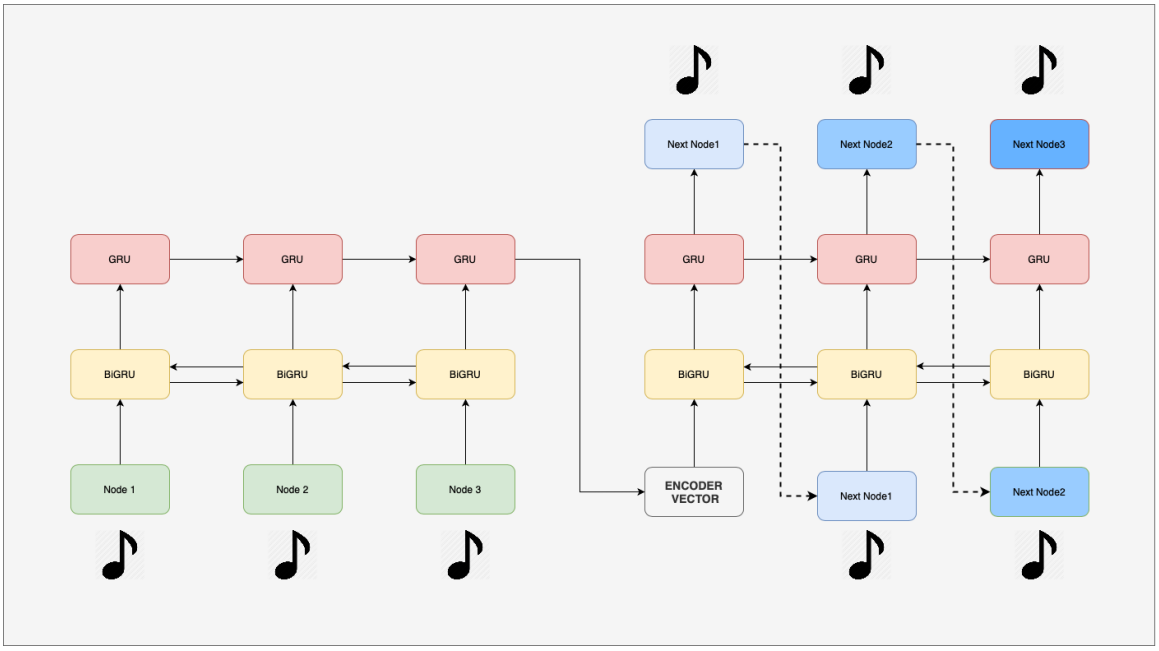

Figure 1: Music encoder-decoder model, with bidirectional GRU for the 1st layer, and GRU for the 2nd layer.

The Machine Learning model was first trained using the training/validation split. Meaning we divided the original data into a split of 90%/10% of data and used the 10% to evaluate the model performance. The model was trained for 30 epochs and was able to perform well on the validation data, the sparse categorical cross-entropy loss difference of the training data, and the validation data is around 0.1. Figure 2 shows the loss/epoch of this step.

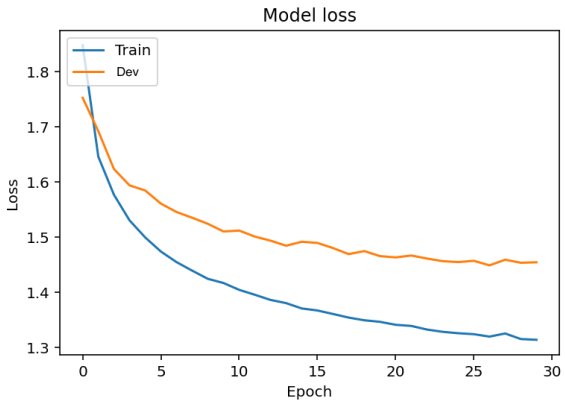

Figure 2: This plot shows the train/dev loss.

After defining the model architecture and the hyperparameters, we made use of all the data, by combining both training and validation as input data, and training (fine-tuning) on all data for 60 more epochs, we also monitored the model loss function on the test data. As shown in the

Figure 3, the test loss is less than the training loss.

To investigate the source of this abnormality, we checked the distribution of the training and the test sets. We plotted the distribution of node occurrence for each data set (log form). As you can see from Figure 4, the distributions are similar, but the testing data has more nodes that did not appear in the dataset.

Lastly we defined temperature for our generative model. Temperature is a parameter used for sampling in the last layer of the neural network. You can think of it as controlling randomness: higher values produce more variation and sometimes even chaos, while lower values are more conservative in their predictions.

References

[1] Nicolas Boulanger-Lewandowsk, Pascal Vincent YoshuaBengio, Patrick Gray, and Chinmaya Naguri. Modeling temporal dependencies in high-dimensional sequences.

[2] Curtis Hawthorne, Erich Elsen, Jialin Song, Adam Roberts, Ian Simon, Colin Raffel, Jesse Engel, Sageev Oore, and Douglas Eck. Onsets and frames: Dual-objective piano transcription. *arXiv preprint arXiv:1710.11153*, 2017.

[3] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Curtis Hawthorne, Andrew M Dai, Matthew D Hoffman, and Douglas Eck. Music transformer: Generating music with long-term structure. *arXiv preprint arXiv:1809.04281*, 2018.

[4] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

[5] Christine Payne. Musenet. openai.com/blog/musenet, 2019.

[6] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.

[7] Bernd Krueger. Classical Piano Midi Page http://www.piano-midi.de, 2018.

[8] Multiple contributions MIDIWORLD https://www.midiworld.com/, 2009

[9] Ian Simon and Sageev Oore. Performance rnn: Generating music with expressive timing and dynamics, 2017.

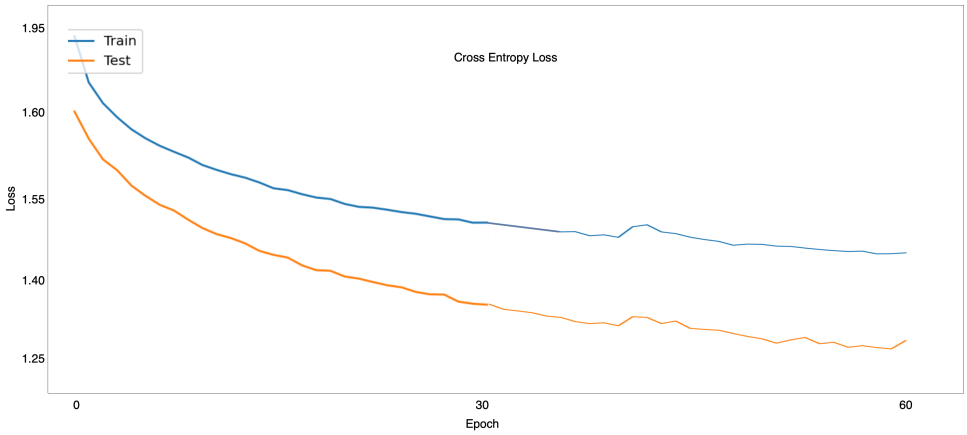

Figure 3: This plot shows the train/test loss for the fine-tuning step. In this step we used all of our training data and ran it for 60 more epochs. You can also see the test loss on this plot

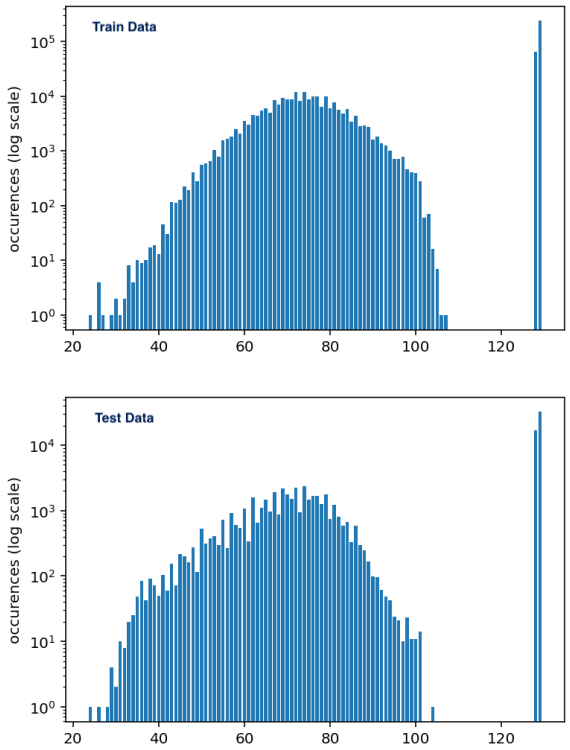

Figure 4: This plots shows the node distribution in each of the datasets. (in log scale)