Music Generation using LSTM networks

Speech Signal Processing Project

Nived Rajaraman

Department of Electrical Engineering
EE14B040

23^{rd} January, 2018

Outline

Outline

Temporal Structure of Music

Music is a fluid art-form. But most pleasant melodies abide by the traditionally studied temporal model of music of *bars* and *subdivision*.

Temporal Structure of Music

Music is a fluid art-form. But most pleasant melodies abide by the traditionally studied temporal model of music of *bars* and *subdivision*. A brief walk through some music theory...

- Note
- Beat
- Bar (Measure)
- Tempo

Outline

1. Introduction
   - Temporal Structure of Music
   - The Music Generation Problem
2. Technical Details
   - Dataset
   - Dataset Preprocessing
3. LSTM architecture
   - LSTM Architecture
   - Dataset Preprocessing
4. Results

The Music Generation Problem

- The music generation problem is to learn the musical structure behind musical compositions and attempt to generate new music, optionally based on a context to the preceding bars.
- We assumed the standard model of the music presented above and attempted to **generate a bar of music conditioned on the previous bar of music** for a single instrument **symbolically** (and not from the audio waveform).

Outline

Dataset

- Midinet Dataset - published in the numpy **npz format** generated via the state-of-the-art dynamic time warping algorithm. Estimated labelling error - 4%

Dataset

- Midinet Dataset - published in the numpy **npz format** generated via the state-of-the-art dynamic time warping algorithm. Estimated labelling error - 4%
- The dataset contains annotations of music compositions from over 330 classical music pieces in violin, piano and bass. The over 1 million annotated labels indicate the precise time of each note in every recording,the instrument that plays each note, and the notes position in the metrical structure of the composition.

Dataset

- Midinet Dataset - published in the numpy **npz format** generated via the state-of-the-art dynamic time warping algorithm. Estimated labelling error - 4%
- The dataset contains annotations of music compositions from over 330 classical music pieces in violin, piano and bass. The over 1 million annotated labels indicate the precise time of each note in every recording,the instrument that plays each note, and the notes position in the metrical structure of the composition.
- The dataset is stored hierarchically as an interval tree - which when queried with a timepoint provides the annotations - which note is being played by each instrument at that moment.

Outline

Dataset Preprocessing

- The dataset is processed into the now commonly used for symbolic music generation **piano-roll format**.

Dataset Preprocessing

- The dataset is processed into the now commonly used for symbolic music generation **piano-roll format**.
- This generates the music as bars, each split into $\mathcal{S}$ number of note locations (which is a hyperparameter). A quarter note is sustained for $\frac{\mathcal{S}}{4}$ notes.

Dataset Preprocessing

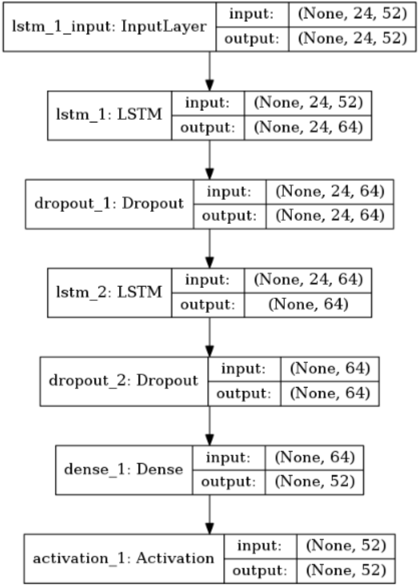
- The dataset is processed into the now commonly used for symbolic music generation **piano-roll format**.
- This generates the music as bars, each split into $\mathcal{S}$ number of note locations (which is a hyperparameter). A quarter note is sustained for $\frac{\mathcal{S}}{4}$ notes.
- For each of the $\mathcal{S}$ locations in a bar, a 128-long 1-or-more hot vector is associated, and longer duration notes are spread across multiple such vectors at the same index (frequency).

Dataset Preprocessing

- The dataset is processed into the now commonly used for symbolic music generation **piano-roll format**.
- This generates the music as bars, each split into $\mathcal{S}$ number of note locations (which is a hyperparameter). A quarter note is sustained for $\frac{\mathcal{S}}{4}$ notes.
- For each of the $\mathcal{S}$ locations in a bar, a 128-long 1-or-more hot vector is associated, and longer duration notes are spread across multiple such vectors at the same index (frequency).
- The generated piano-roll is clipped to lie between the largest and smallest pitch being played to dampen the RAM requirement. This cuts the memory requirement roughly in half.

Outline

LSTM Architecture



LSTM Architecture

- While a Generative model might be a good option to generate compositions, in this project the distribution of notes was modelled using an LSTM and sampled from the output layer to generate an estimate for the composition has been looked at.
- The network hyperparameters were decided by picking the net having the maximum size and varying the dropout parameters until overfitting was minimized[1].

LSTM Architecture

- While a Generative model might be a good option to generate compositions, in this project the distribution of notes was modelled using an LSTM and sampled from the output layer to generate an estimate for the composition has been looked at.
- The network hyperparameters were decided by picking the net having the maximum size and varying the dropout parameters until overfitting was minimized[1].
- Thus the optimal network after several re-trainings was found to have 2 hidden layers with 256-LSTM cells in each. A dropout layer with probability 0.5 is added to prevent the network from overfitting on the training set. The recurrent activations for neurons was taken as the sigmoid function, and the neuron output activation was taken as a hyperbolic-tan function.

Outline

Dataset Preprocessing

The other hyperparameters of the network were chosen as below:

1. Batch Size: 128
2. Number of epochs: 20 (training time of 25 minutes per epoch)
3. Training/Validation split: 80/20
4. Loss function optimizer: Adam Optimizer with Nesterov Momentum[6]

Network Architecture

- The output layer terminates in sigmoid's as the intention is to judge the existence of each pitch in the next node independent from the others. While this may not be the best choice due to interdependency between notes, it certainly avoids the case of unintentionally killing the probabilities of presence of other nodes due to a highly activated output neuron (softmax classifier).

Network Architecture

- The output layer terminates in sigmoid's as the intention is to judge the existence of each pitch in the next node independent from the others. While this may not be the best choice due to interdependency between notes, it certainly avoids the case of unintentionally killing the probabilities of presence of other nodes due to a highly activated output neuron (softmax classifier).
- The loss function that optimized while training the network is the **binary cross entropy loss**. Since we are dealing with a case where multiple outputs neurons may be activated for a given input, binary classification was the choice. This is more forgiving when an output is activated when in fact it should not be.

We also introduced a modification of a Hamming-distance like metric to govern the true accuracy of the network, which is illustrated by the below expression:

$$\text{Metric} = \frac{1}{N}\sum_{i=1}^{N} \text{abs}(y_{\text{true}}) - \text{abs}(y_{\text{true}} - \text{round}(y_{\text{pred}}))$$

$$\text{Where, } N = \#\ \text{MIDI-frequencies}$$

The metric appreciates the network outputting all-zeros after rounding. But the payoff for a correct prediction increases linearly with the number of notes at any instance. So the network might *immediately learn to start outputting all-zeros* as this would allow it to tackle and learn the easy cases of silence being the correct prediction, but as the number of epochs increase, it would (hopefully) learn that predicting notes correctly would generate better pay-offs in the long run.

Results

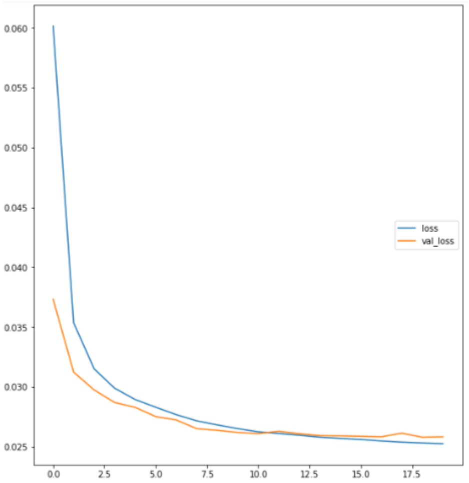


Figure: Training and Validation Loss vs. epochs

Results

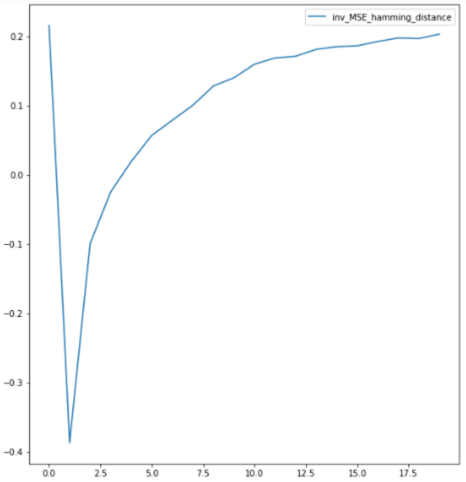


Figure: Accuracy Metric vs. epochs

Results

A sample music composition generated by the network:

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, Koray Kavukcuoglu, *WaveNet: A Generative Model for Raw Audio*, (2016)

Li-Chia Yang, Szu-Yu Chou, Yi-Hsuan Yang, *MidiNet: A Convolutional Generative Adversarial Network for Symbolic-domain Music Generation*, ISMIR (2017)

John Thickstun, Zaid Harchaoui, Sham Kakade, *Learning Features of Music from Scratch*, (2016)

Google, *Melody RNN Model*, https://github.com/tensorflow/magenta/tree/master/magenta/model

References II

Andrej Karpathy, *The Unreasonable Effectiveness of Recurrent Neural Networks*, http://karpathy.github.io/2015/05/21/rnn-effectiveness/ (2015)

Soumith Chintala, Emily Denton, Martin Arjovsky, *How to Train a GAN? Tips and tricks to make GANs work*, https://github.com/soumith/ganhacks