Deep Jammer: A Music Generation Model

Justin Svegliato, Sam Witty

College of Information and Computer Science
University of Massachusetts, Amherst

Introduction

For this project, we explored the use of deep learning methods to generate music. In particular, we

- developed a novel neural network architecture,
- trained various configurations of the neural network on a dataset of classical music,
- transferred learned representations from the classical dataset to a jazz dataset,
- generated a collection of musical segments for the various network configurations, and
- surveyed a sample of peers to quantitatively assess the effectiveness of the various configurations.

Model

The input is a segment of a musical piece, which is a matrix of 128 time steps, 88 notes, and 78 attributes. Each attribute is defined below:

- **Position**: The piano key position.
- **Pitch Class**: A categorical array of pitches.
- **Vicinity**: An array of neighboring note states.
- **Beat**: The location in a measure.

The output is a matrix of 88 notes and 2 predictions:

- **Play Probability**: The probability of the note being played.
- **Articulate Probability**: The probability of the note being held.

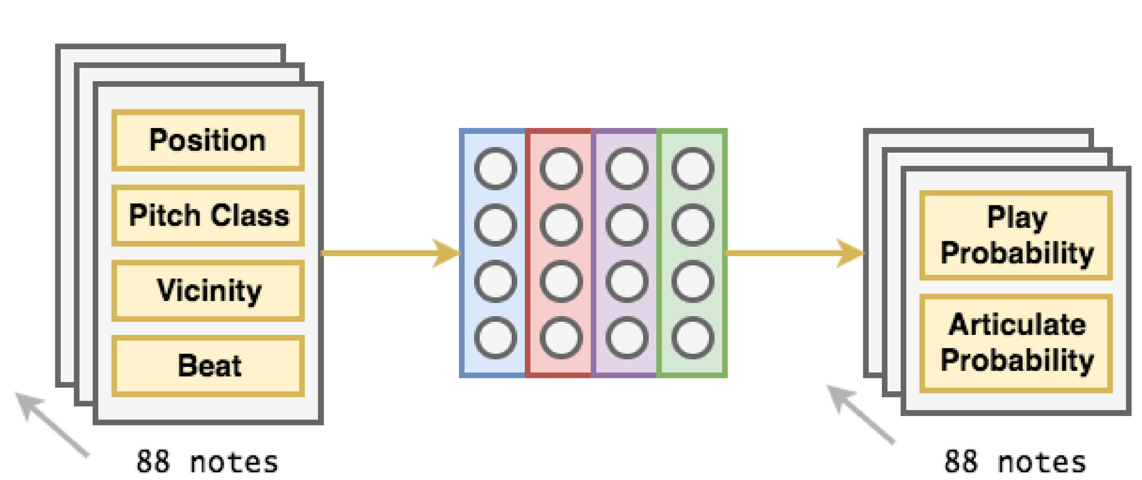

Figure 1: An overview of the music generation model.

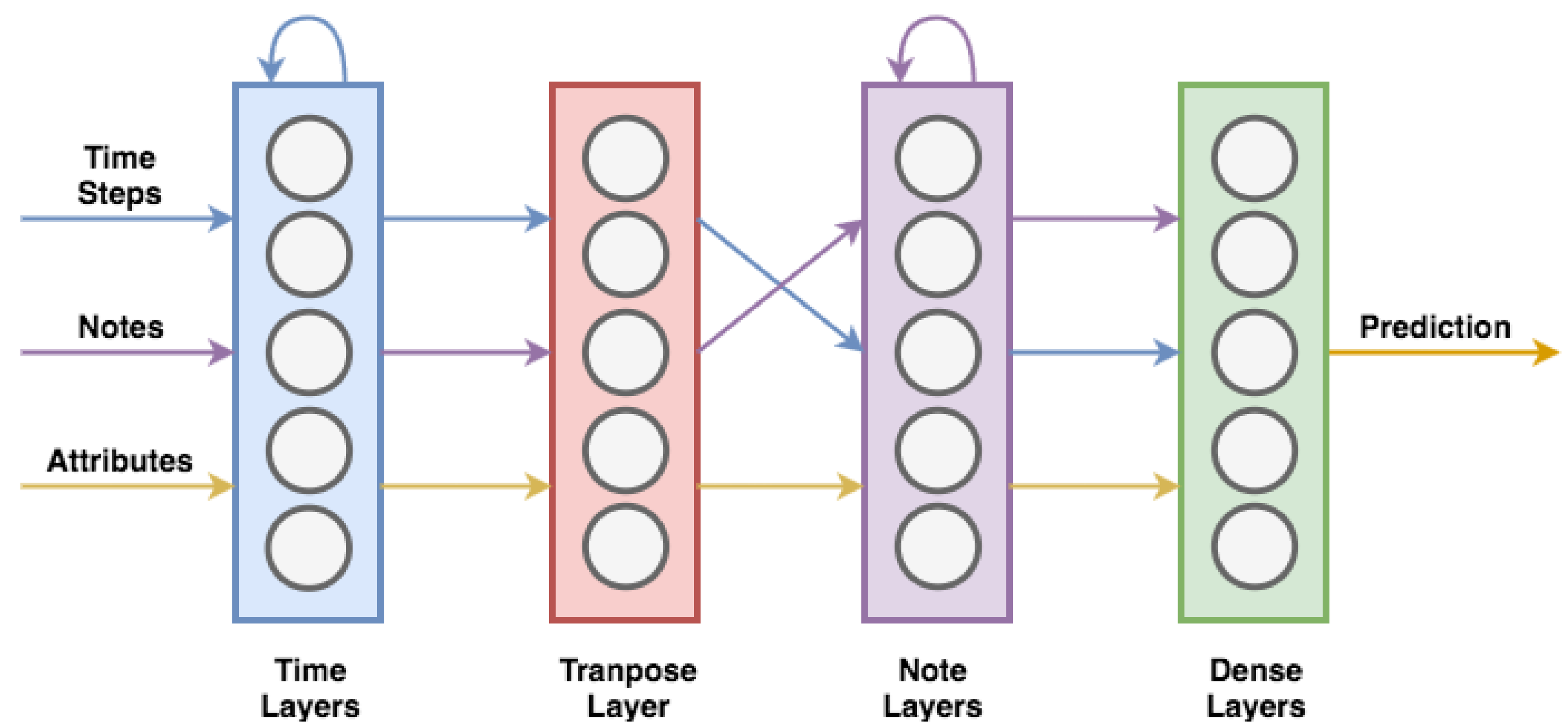

Figure 2: The neural network architecture of the music generation model.

Training

Training is conducted for both classical and jazz music as follows:

- **Feed** in *all* time steps of a random segment.
- **Predict** *all* time steps as output.
- **Update** the model weights using Adadelta.
- **Repeat** the process by feeding in the next *segment*.

The loss is defined as the negative log-likelihood of the model given the observed data:

$$loss = -\sum_{t,n} \log\left(\sigma_{t,n} y_{t,n} + (1 - \sigma_{t,n})(1 - y_{t,n})\right)$$

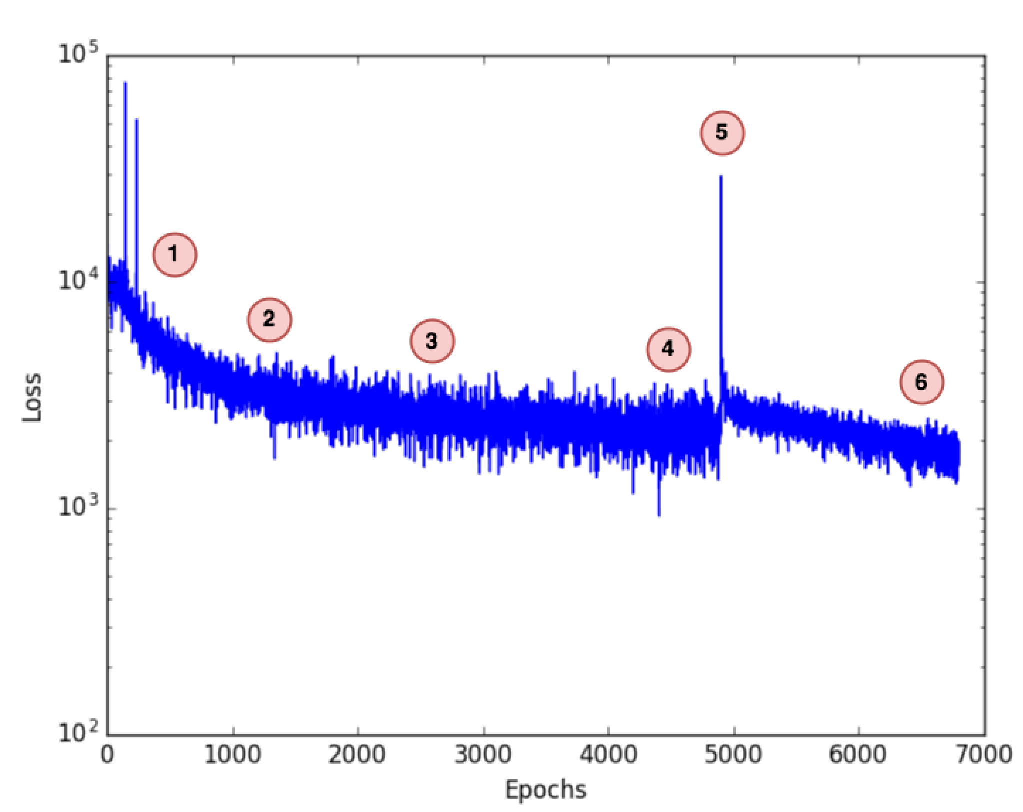

Figure 3: The loss curve of the Large Network with Fine-Tuning.

Generation

Generation is slightly different from the training process:

- **Feed** in a *single* time step of a random segment.
- **Predict** the *next* time step as output.
- **Select** the notes to be played.
- **Repeat** the process by feeding in the *next* time step.

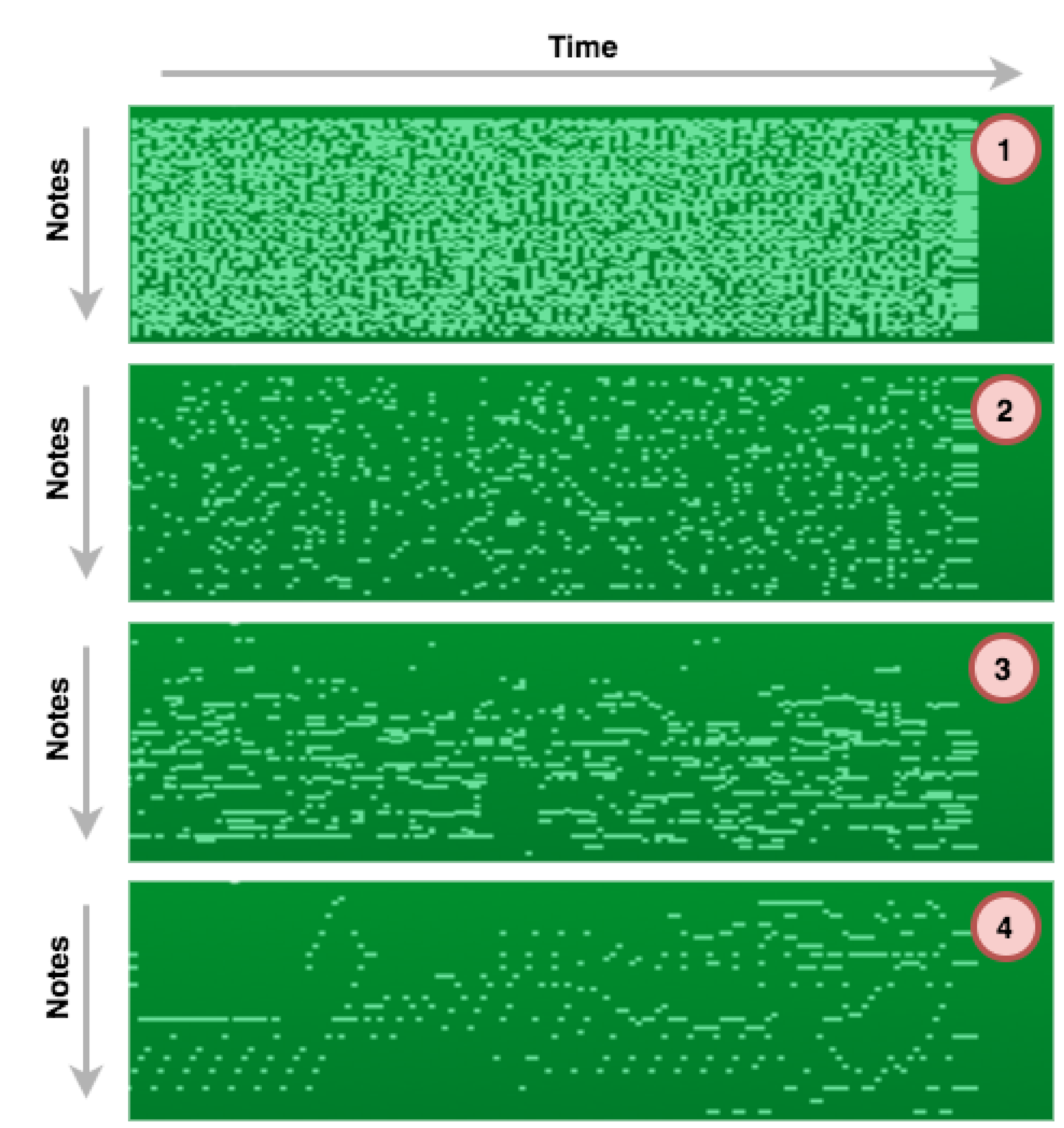

Figure 4: Music at various training stages for the Large Network.

Experiments

A variety of experiments were conducted to test the effects of network size and dropout.

Table 1: The hyperparameters used in the experiments.

Parameter	Large	Medium	Small	Dropout
Time Layer 1 Size	300	150	75	300
Time Layer 2 Size	300	150	75	300
Note Layer 1 Size	100	50	25	100
Note Layer 2 Size	50	25	13	50
Dense Layer Size	2	2	2	2
Dropout Strength	0	0	0	0.5

Results

We conducted an online survey where 26 participants assessed 14 generated segments and 2 real segments.

Table 2: The results of the survey.

Model	Rating	Believability
Large Network	6.7	76
Medium Network	6.2	73
Small Network	6.1	67
Large Network with Dropout	6.0	69
Extended Training	4.3	40
Fine-Tuning	4.7	48
Real Classical	8.1	100
Real Jazz	7.3	92

References

[1] Conklin, D. *Chord Sequence Generation with Semiotic Patterns*. Journal of Mathematics and Music, 10 (2):92-106, 2016.

[2] Johnson, D. *Composing Music with Recurrent Neural Networks*. Hexahedria, 2016.

[3] Liu, I. and Ramakrishnan, B. *Bach in 2014: Music Composition with Recurrent Neural Network*. Arxiv, 2014.