Modeling the Latent Space of Symbolic String Quartet Music

Alex Kyllo
akyllo@uw.edu
University of Washington
Bothell, WA, USA

ABSTRACT

This paper presents the results of a project to apply a recurrent variational autoencoder to the task of modeling a latent space for generating pieces of multi-instrument symbolic classical music through interpolation. The code utilized for this research is available at: https://github.com/jackPhn/CSS586_Project/

KEYWORDS

deep learning, recurrent neural networks, sequential models, music modeling, latent space modeling, generative modeling

1 INTRODUCTION

Machine learning models of music have interesting applications in music information retrieval and creative tools for musical artists and educators. Generative models can create accompaniments for music, blend and transfer styles between clips of music, and even generate entirely new music. Music is challenging to model because it is inherently high-dimensional, exhibiting a complex hierarchy of recurring patterns and long-range temporal dependencies, and because musical scores have multiple possible digital representations with distinct advantages and disadvantages.

Depending on the task, machine learning models of music may be trained on the audio signal of a musical performance, either in a time domain or a frequency domain representation, or they may be trained on a digital symbolic representation of music, the most common of which is MIDI (Musical Instrument Digital Interface) notation. MIDI is an encoding of music as streams of bytes in one or more tracks or channels, each representing a sequence of 128 possible pitch values (where 0 is the lowest pitch and 127 is the highest), along with timing, pressure and instrument identifier values. A generative symbolic music model can produce a symbolic score in MIDI format, which must be played by synthesizer software or by humans to produce an audio music performance. This project focuses on generative modeling of symbolic (MIDI) music to compose original musical scores by blending input scores through continuous latent space interpolation.

A survey of related works in generative modeling of polyphonic music turned up several models focused on multi-instrument pop/rock music arrangements or classical piano performances. In contrast, this project focuses on variational autoencoding and latent space interpolation of classical string quartets (arrangements for an ensemble of two violins, one viola and one cello) and is, to the author's knowledge, the first project to apply this modeling approach to this type of musical arrangement.

2 RELATED WORK

The state of the art in music generation still has a long way to go before it can consistently generate music scores or performances that would be enjoyable and popular for humans to listen to, but for this reason it is an area of significant opportunity, where a number of recent research projects have shown promising progress.

Google's Magenta is an umbrella project for music deep learning research and development of software tools to expose these models for use by creative artists and students.

MusicVAE, part of the Magenta project, is a variational Long Short-Term Memory (LSTM) autoencoder for MIDI that incorporates a novel hierarchical structure using a "conductor" recurrent layer in its decoder model to better capture structure at multiple levels and avoid the problem of "posterior/mode collapse" whereby a generative model learns to ignore its latent codes and rely on autoregression to generate output sequences [13]. This model is trained on 16-bar paragraphs of music and is capable of generating new melodies that blend two given melodies together via latent space interpolation. Among the literature surveyed, MusicVAE's approach is the most similar to this work; in comparison MusicVAE dedicates significant effort to devising an architecture that allows the model to generate very long sequences of notes, which is beneficial for other generation tasks but not strictly necessary for interpolation. Our model is significantly simpler and demonstrates that interpolation works well even on a single bar of music at a time.

Another Magenta model called Music Transformer is a generative model that borrows its approach from the Natural Language Processing (NLP) domain, using a self-attention network to model MIDI music as a sequence of discrete tokens with relative positional dependencies [5]. The focus of this model is on learning long-term dependencies in music to produce longer clips of music with coherent structure. Music Transformer was trained on a dataset of Piano-e-competition performances [4] and its generated piano music received favorable qualitative (Likert scale) ratings from human listeners for its resemblance to human-composed music [5]. In contrast to MusicVAE and this work, Music Transformer is capable of generating much longer sequences of music, but does so via sequence extrapolation rather than interpolation, so it can continue a given priming melody but does not blend multiple given melodies together. It also assumes a single instrument track (piano).

MuseGAN [3] is an application of Generative Adversarial Networks (GAN) to polyphonic MIDI music generation, trained on four-bar phrases of a multi-track pianoroll representation of rock songs from the Lakh Midi Dataset [11]. Like MusicVAE, MuseGAN includes a two-level generator that first samples latent codes at the phrase or bar level, then generates notes within the bars, to produce longer-term structural patterns.

A major advantage of working with the symbolic representation of music is that it is of far lower dimensionality than the raw audio waveforms of a recorded performance, which makes it less computationally expensive. However, there are many stylistic aspects

of musical performance that are not captured by a symbolic representation, and may be specific to a particular performer, so the expressiveness of symbolic generative models is limited in comparison [7].

Other research has focused on modeling raw audio waveforms directly. WaveNet is a causal convolutional neural network for generating raw audio waveforms, developed by Google DeepMind, which achieves state of the art performance in generating natural sounding speech from text, but is also capable of generating short, realistic snippets of audio music [9]. Another model named SampleRNN generates raw audio waveforms using a three-tier hierarchy of gated recurrent units (GRU) to model recurrent structure at multiple temporal resolutions [8].

Prior work points out that the division between symbolic music notes and music performances, is analogous to the division between symbolic language and utterances in speech, which may inspire ideas for combining the two approaches [4]. A paper from Boston University describes an effort to combine the symbolic and waveform approaches to music modeling, by training an LSTM to learn melodic structure of different styles of music, then providing generations from this model as conditioning inputs to a WaveNet-based raw audio generator [7].

While there may be significant opportunity in future approaches that combining the two, contemporary research generally treats symbolic and audio music modeling as separate problem domains and this work does the same, choosing to focus on symbolic music.

3 METHODS

3.1 Datasets

The dataset used for this research project is MusicNet, which is a collection of 330 freely licensed European classical music recordings with aligned MIDI scores [14]. The model is trained on 36 string quartets (four-part arrangements with two violins, one viola and one cello) by composers Bach, Beethoven, Dvorak, Haydn, Mozart, and Ravel.

3.2 Data Preprocessing

Several choices must be made in how to preprocess binary MIDI files into training examples for a neural network. There are multiple open-source Python packages that assist with the process of reading MIDI files from their binary on-disk representations into Python objects, such as pretty_midi [12], Pypianoroll [2] and music21 [1].

In order to accommodate polyphonic music, we each MIDI file is converted into a modified pianoroll representation, an example of which is visualized in Figure 2. While the standard pianoroll represents each time-step as a multi-hot encoded vector of note values, because most notes are not being played at most timesteps there is an extreme class imbalance and sparsity in the matrix. This is reported in [6] to cause problems in model fitting and metrics calculation, so they introduce an alternative "multi-stream" representation of the pianoroll, which we closely, (but not exactly) emulate in this work.

In our modified representation, each instrument track is a sequence of integers from 0-129 representing the MIDI note pitch value being played at the given time step, where values 0-127 are pitches, 128 is a rest (silence) and 129 is sustain, indicating that the previously played pitch is held continuously. MIDI data contains pressure values that indicate how hard or soft each note is played; this information is discarded in our preprocessing and all notes are represented with constant pressure. Because chords (multiple notes being played simultaneously by the same instrument) are relatively rare in string quartets, the lowest note of each chord is taken and the rest discarded, to reduce dimensionality. Because notes at the extreme low and high end are rare and are out of range of common instruments, the note pitch values are clipped to a narrower range and ordinal encoded. For the 36 string quartets found in the MusicNet database, this results in 66 distinct values, where 0-63 represent notes played by the violins, viola and cello, 64 is rest and 65 is sustain. Figure 1 depicts the distribution of these pitch values per instrument. Our software package also includes functions to convert this representation back into a Music21 Score object, which can then be written to a MIDI file.

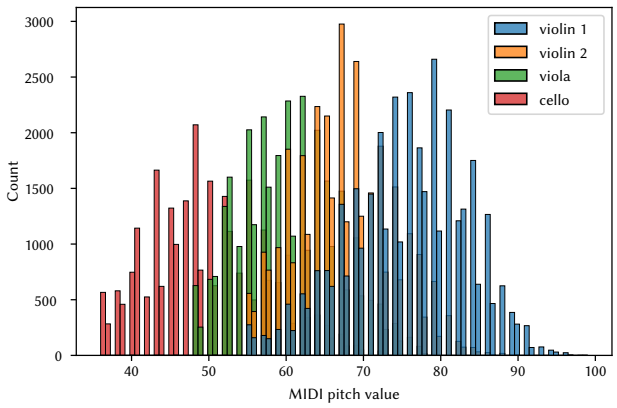

Figure 1: Distribution of pitch values per instrument in the MusicNet string quartets

Because songs are typically at least a few minutes long and of varying length, it will not be feasible to train with entire songs as examples, so we will crop songs into phrases of equal numbers of beats to use as training data.

The result of this preprocessing is that each training example isbe a 2D matrix of shape (time steps x instrument tracks) and stacking the training examples produces a 3D tensor. Information regarding which track corresponds to which musical instrument is lost and therefore must be maintained separately to reconstruct the MIDI representation. For string quartets this instrument ensemble is always `[40, 40, 41, 42]`, which is an array of MIDI instrument codes representing two violins, one viola and one cello. Tempo information is also lost from this representation, so the outputs are normalized to the default tempo of 120 beats per minute.

As the model will produce one output per instrument track, it can incorporate only a fixed selection of instrument parts, similar to how MusicVAE models three-part (drum, bass and melody) [13] and MuseGAN models five-part (drum, bass, guitar, string, piano)

arrangements. Therefore this model is purpose-built for string quartets, which are the most common type of arrangement found in the MusicNet dataset.

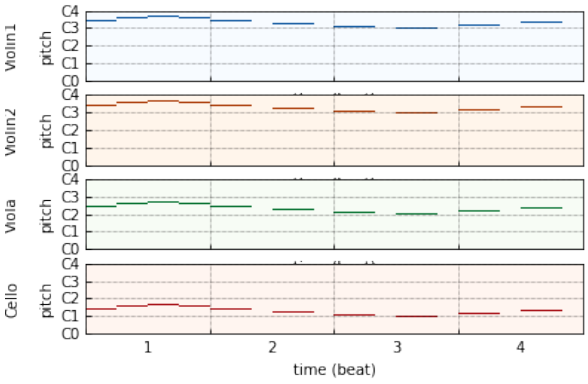

Figure 2: Pianoroll visualization of the first measure of Beethoven's Serioso String Quartet in F Minor

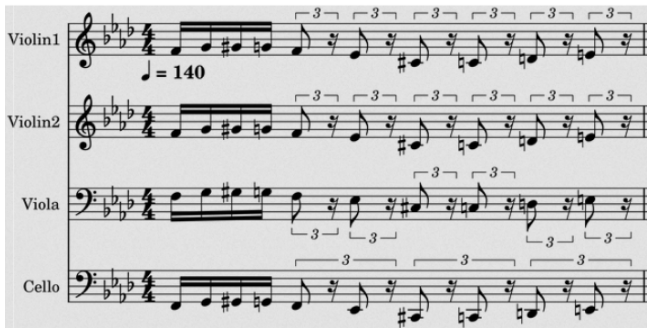

Figure 3: Sheet music of the first measure of Beethoven's Serioso String Quartet in F Minor

3.3 Model Design

The model is a recurrent variational autoencoder; both Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) cell types were tested but this choice did not make a significant difference in the results.

The encoder model utilizes an embedding layer to encode the note pitch integers as vectors; embedding vector length is a tunable hyperparameter and we found the best results with it set to 8. Then two LSTM layers, with a dropout layer in between, convert the sequences into latent codes, which are modeled as a multidimensional Gaussian distribution z, produced by the encoder output Lambda layer (Figure 5).

The decoder model samples from the latent space z with the distribution parameterized by the μ and $log(\sigma)$ values learned by the encoder, and then utilizes two LSTM layers with a dropout layer between them to generate sequences from the latent codes. Then the network splits into four parallel outputs, one per instrument track, where a fully connected layer with softmax activation is used to select the most probable note value for the the current track at the each time step (Figure 5).

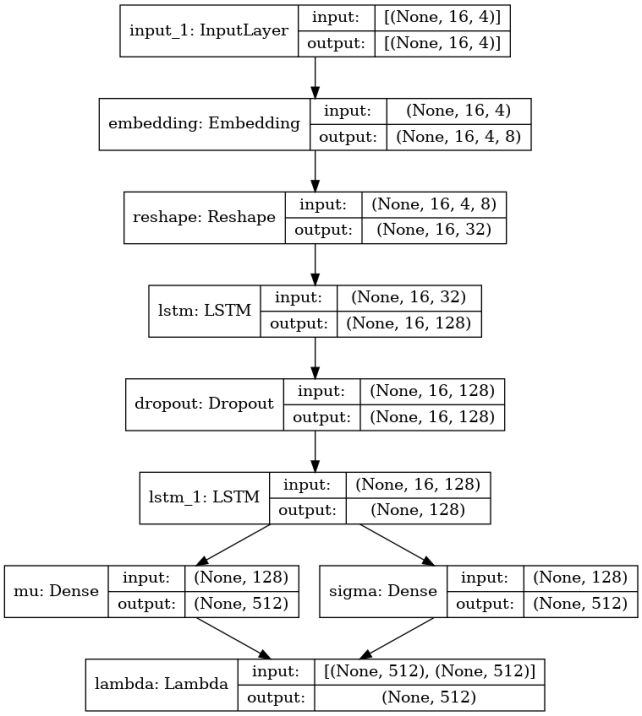

Figure 4: A directed graph diagram of the encoder network.

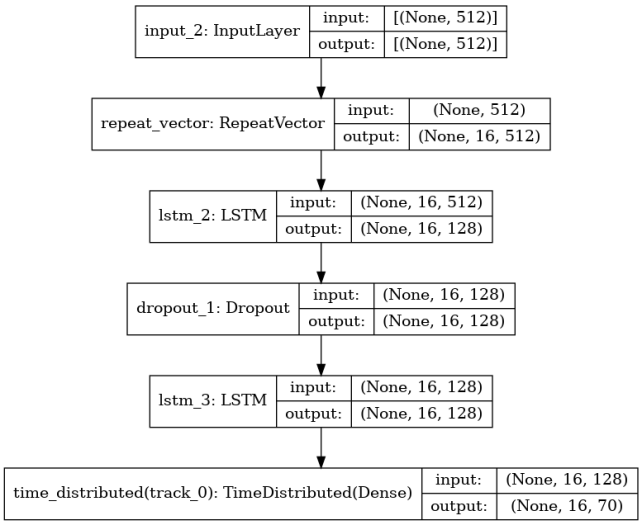

Figure 5: A directed graph diagram of the decoder network (only one of four parallel output layers shown).

The autoencoder reconstruction loss function is sparse categorical cross-entropy, which is added to the Kullback-Leibler divergence

of the approximate posterior distribution from a multivariate Gaussian distribution with mean 0 and standard deviation 1, resulting in the (negative) evidence lower bound (ELBO) loss (Equation 1), which is a standard variational autoencoder loss function. This loss function is applied to each track independently and then summed across all four tracks with equal weight.

$$\mathbb{E}[logp_\theta(x|z)] - \mathbb{KL}(q_\lambda(z|x)||p(z)) \quad (1)$$

where λ and θ represent the encoder and decoder parameters respectively, while $q_\lambda(z|x)$ represents the encoder, which approximates the true posterior probability $p(z|x)$, and $p_\theta(x|z)$ represents the decoder, which parameterizes the likelihood $p(x|z)$ [13].

3.4 Model Experimentation

The following hyperparameters were tuned, with tested values listed and best parameters in **bold**:

- batch size: **32**, 64, 96
- learning rate: 0.001, **0.0002**
- recurrent layer type: LSTM, GRU (made no difference)
- recurrent encoder layer directions: **unidirectional**, bidirectional
- recurrent layer cells: 64, **128**, 256
- latent code dimension: 128, 256, **512**
- embedding dimension: 4, **8**, 16
- dropout rate: 0.2, 0.4, **0.5**
- time steps per beat (quarter note): **4**, 8
- beats per phrase: **4**, 8

Each model was trained for up to 500 epochs with an early stopping patience of 20 epochs. The model was cross-validated on 10% of the training data after each epoch. The best model was selected by a combination of validation loss and listening to hand-picked sample reconstructions and interpolations.

Data augmentation was also tested–[10] suggests augmentation via pitch shifting each training example up or down by up to six semitones in order to create additional training examples and reduce overfitting, so we followed that recommendation and used it to double the size of the training dataset. As a result, we saw much lower validation losses (compare Figure 6 and Figure 7 and a small but noticeable improvement in the fidelity of the song reconstructions, suggesting that this method was effective in reducing overfitting.

4 RESULTS

The reconstructed music produced by the model sounds reminiscent of the original inputs, preserving the key signature and general rhythmic and melodic structure, albeit with noticeable differences at the individual note level.

A collection of samples of the autoencoded outputs and interpolations, converted to mp3 format, is presented at the following URL: https://drive.google.com/drive/folders/1SlQOI_cL452PftO3KmOYO_qglE2RotCX?usp=sharing. Each of the 18 directories contains five interpolations between two of the string quartet pieces in the MusicNet corpus, numbered 0-4, where 0 is the reconstruction of the first piece, 4 is the reconstruciton of the second piece, and 1-3 are linear interpolations between the two pieces. Each file is cropped to the first 16 measures only.

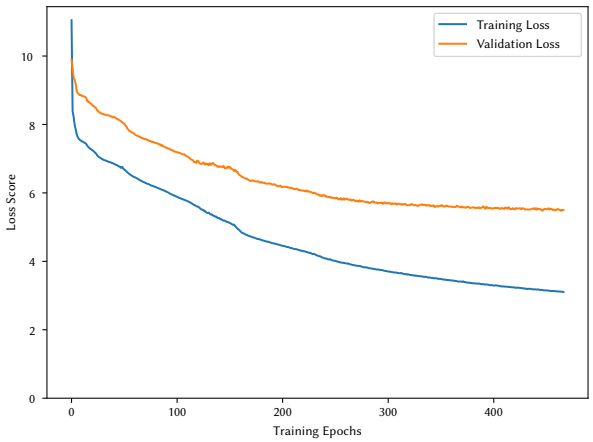

Figure 6: Training and validation loss for the best performing model - without data augmentation

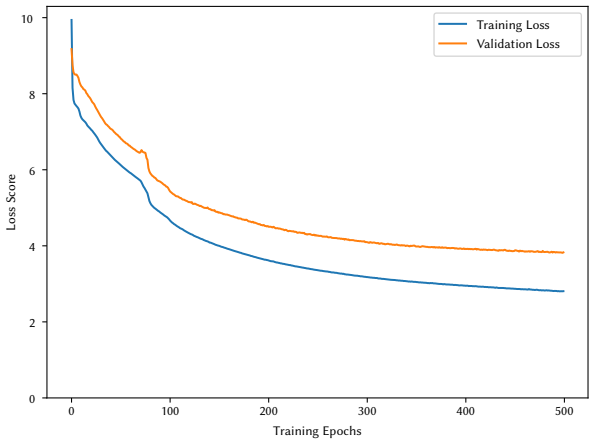

Figure 7: Training and validation loss for the best performing model - with data augmentation

The interpolations are between the following 18 pairs of pieces

- `Haydn/2104_op64n5_1.mid` and `Ravel/2178_gr_rqtf2.mid`
- `Haydn/2105_op64n5_2.mid` and `Ravel/2179_gr_rqtf3.mid`
- `Haydn/2106_op64n5_3.mid` and `Ravel/2177_gr_rqtf1.mid`
- `Beethoven/2497_qt11_4.mid` and `Mozart/1822_kv_421_1.mid`
- `Beethoven/2433_qt16_3.mid` and `Mozart/1859_kv_464_2.mid`
- `Beethoven/2368_qt12_4.mid` and `Mozart/1807_kv_387_3.mid`
- `Beethoven/2314_qt15_2.mid` and `Mozart/1791_kv_465_4.mid`
- `Beethoven/2480_qt05_1.mid` and `Mozart/1792_kv_465_1.mid`
- `Beethoven/2481_qt05_2.mid` and `Mozart/1835_kv_590_3.mid`
- `Beethoven/2379_qt08_4.mid` and `Mozart/1805_kv_387_1.mid`
- `Beethoven/2365_qt12_1.mid` and `Mozart/1793_kv_465_2.mid`
- `Beethoven/2562_qt02_4.mid` and `Mozart/1790_kv_465_3.mid`
- `Beethoven/2494_qt11_1.mid` and `Mozart/1789_kv_465_2.mid`
- `Beethoven/2403_qt01_4.mid` and `Mozart/1788_kv_465_1.mid`

- Beethoven/2376_qt08_1.mid and Dvorak/1916_dvq10m1.mid
- Beethoven/2384_qt13_4.mid and Bach/2242_vs1_2.mid
- Beethoven/2560_qt02_2.mid and Beethoven/2621_qt07_1.mid
- Beethoven/2377_qt08_2.mid and Beethoven/2381_qt13_1.mid

4.1 Model Evaluation

Evaluation of generative models is challenging because there is no equivalent of an accuracy metric like what is used in supervised learning. For autoencoder models we can measure how accurately the model can reconstruct its own inputs, but this does not tell us the quality of the interpolated examples. Generative models are typically evaluated using a combination of qualitative metrics whereby human judges rate the quality of the generated examples (essentially a Turing test), and quantitative metrics that assess the differences in the parametric distributions of generated and real examples. Yang and Lerch (2020) proposes a set of metrics informed by music theory, for probabilistically evaluating how similar the generations are to known sample distributions of real music [15]. These metrics include counts, ranges, histograms and transition matrices of pitches and note lengths, then the Kullback-Leibler divergence and overlapping area of the probability density functions are used to compare against known reference distributions per musical genre [15]. Due to the cost and time requirements associated with designing a human subjects experiment, we utilize this quantitative approach to the quality assessment of generated samples.

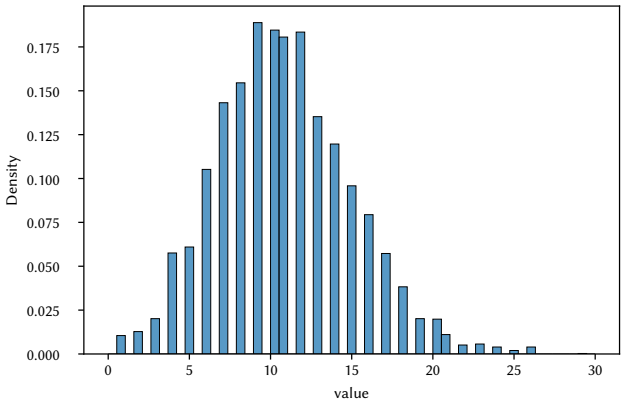

Figure 8: Distinct pitch counts per sample for source samples.

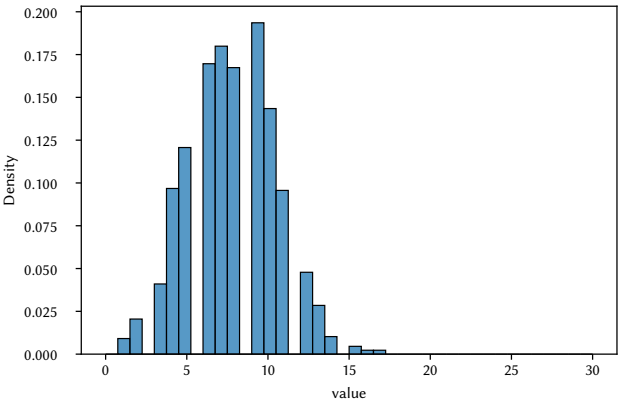

Figure 9: Distinct pitch counts per sample for generated samples.

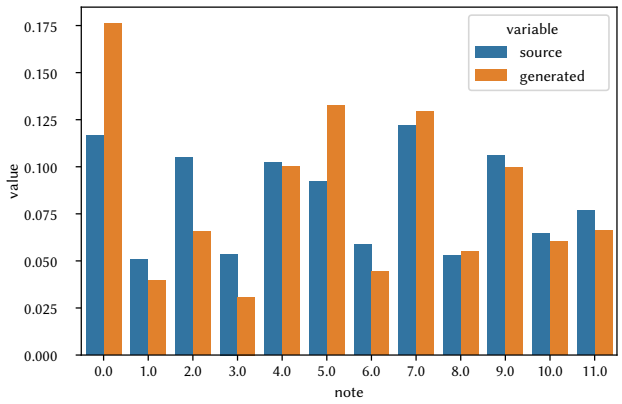

Figure 10: Comparative pitch class histogram for source and generated samples.

5 DISCUSSION

REFERENCES

[1] Michael Scott Cuthbert and Christopher Ariza. [n.d.]. music21: A Toolkit for Computer-Aided Musicology and Symbolic Music Data. ([n. d.]), 6.

[2] Hao-Wen Dong and Wen-Yi Hsiao. 2018. Pypianoroll: Open Source Python Package for Handling Multitrack Pianoroll. (2018), 2.

[3] Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. 2017. MuseGAN: Multi-track Sequential Generative Adversarial Networks for Symbolic Music Generation and Accompaniment. arXiv:1709.06298 [eess.AS]

[4] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. 2019. Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset. arXiv:1810.12247 [cs.SD]

[5] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Ian Simon, Curtis Hawthorne, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck. 2018. Music Transformer. (2018). arXiv:1809.04281 http://arxiv.org/abs/1809.04281

[6] Harish Kumar and Balaraman Ravindran. 2019. Polyphonic Music Composition with LSTM Neural Networks and Reinforcement Learning. arXiv:1902.01973 [cs.SD]

[7] Rachel Manzelli, Vijay Thakkar, Ali Siahkamari, and Brian Kulis. 2018. Conditioning Deep Generative Raw Audio Models for Structured Automatic Music. *CoRR* abs/1806.09905 (2018). arXiv:1806.09905 http://arxiv.org/abs/1806.09905

[8] Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron Courville, and Yoshua Bengio. 2017. SampleRNN: An Unconditional End-to-End Neural Audio Generation Model. (2017). arXiv:1612.07837 http://arxiv.org/abs/1612.07837

[9] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. WaveNet: A Generative Model for Raw Audio. (2016). arXiv:1609.03499

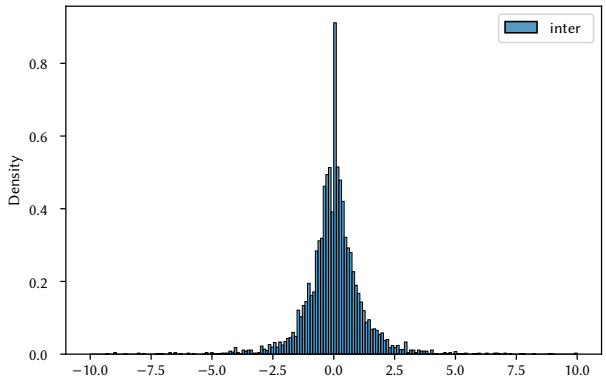

Figure 13: Average pitch interval per sample for source samples.

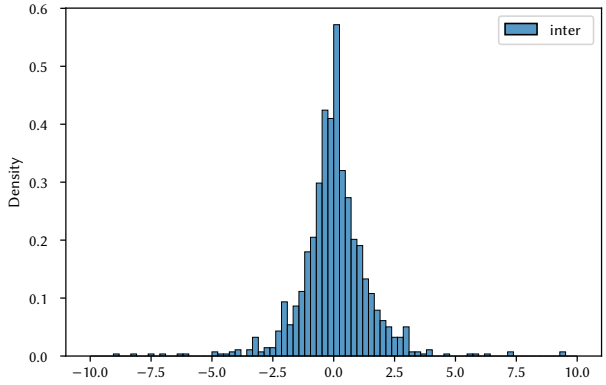

Figure 14: Average pitch interval per sample for generated samples.

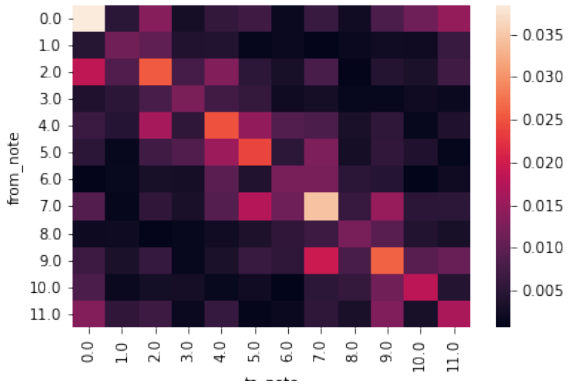

Figure 15: Pitch class transition matrix for source samples.

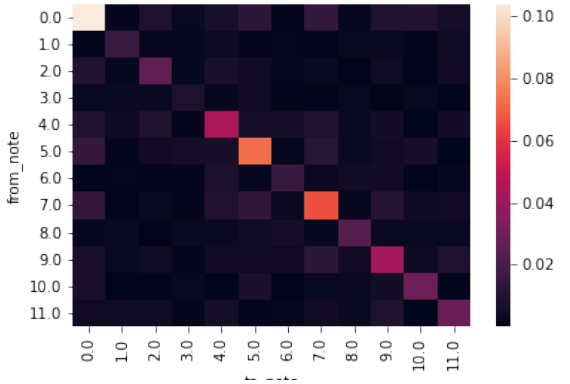

Figure 16: Pitch class transition matrix for generated samples.

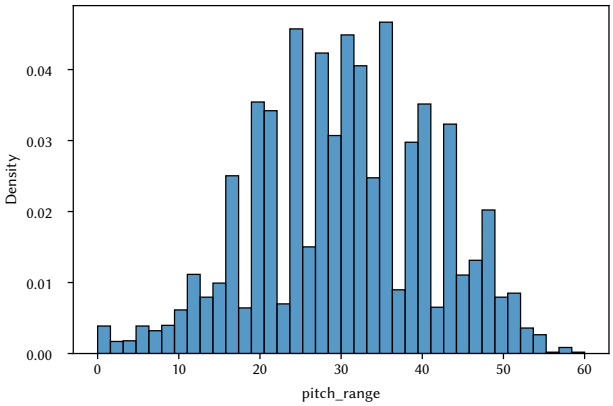

Figure 11: Pitch range per sample for source samples.

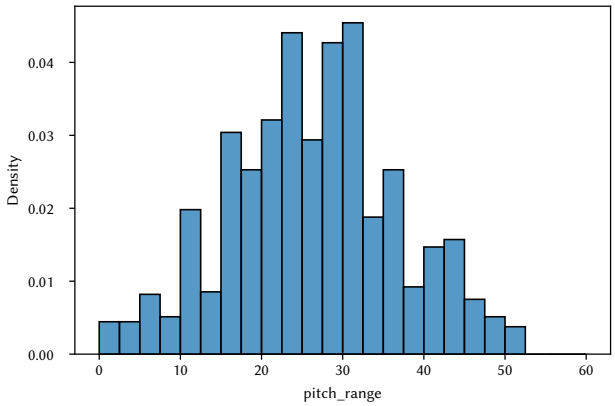

Figure 12: Pitch range per sample for generated samples.

http://arxiv.org/abs/1609.03499

[10] Sageev Oore, Ian Simon, Sander Dieleman, Douglas Eck, and Karen Simonyan. 2018. This Time with Feeling: Learning Expressive Musical Performance. (2018). arXiv:1808.03715 http://arxiv.org/abs/1808.03715

[11] Colin Raffel. 2016. Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching.

[12] Colin Raffel and Daniel P.W. Ellis. 2018. Intuitive Analysis, Creation and Manipulation of MIDI Data With pretty_midi. (2018), 2.

[13] Adam Roberts, Jesse Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. 2018. A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music. In *Proceedings of the35th International Conference on Machine Learning*. 10.

[14] John Thickstun, Zaid Harchaoui, and Sham Kakade. 2017. Learning Features of Music from Scratch. arXiv:1611.09827 [stat.ML]

[15] Li-Chia Yang and Alexander Lerch. 2020. On the evaluation of generative models in music. 32, 9 (2020), 4773–4784. https://doi.org/10.1007/s00521-018-3849-7