Measuring Music Quality Using JETSING (JET Serialism Is Not Good)

Jonathan Sabini **Ethan Fleming** **Taylor Noah**

Portland State University, 2022

Abstract

The most advanced music generation models to date unsurprisingly use musical theory to work effectively. However, the metrics used to score these music generation models are intended for generic machine learning purposes. Often times this leads to many models sounding dramatically different, while having scores that are indistinguishable. Additionally these metrics are dependent on a model's internal weights, rather than scoring the model for its music alone. Shouldn't the metric scoring the models also be designed with music theory in mind?

1 Introduction

Music is inherently repetitive, reusing motifs to form a compelling melody that's self-coherent throughout the piece. Many current models for music generation often have no notion of a long-term structure in musical composition and sound, as though the computer musician is wandering around unsure how to compose a consistent rhythm. The Music Transformer model (Huang et al., 2018) uses close attention to overcome this issue, maintaining long-range coherence throughout its generated musical pieces.

Although the music sounds drastically better than models that don't use relative self-attention, the negative log-likelihood metric's numerical results don't demonstrate this difference well. Many metrics, including the negative log-likelihood, can only be utilized on a machine learning model and wouldn't be able to judge a stand-alone music piece. In general, from what we have found, most metrics used to evaluate music generation models have little foundation on musical theory.

2 Related Work

Countless papers have been written on the topic of music generation, but many of them only use an understanding of musical theory to improve the models themselves. (Huang et al., 2018) worked with Vaswani to utilize the self-attention mechanisms of their original transformer model from (Vaswani et al., 2017), but adjusting it for music generation. (Kotecha and Young, 2018) used a bi-axial LSTM to generate polyphonic music aligned with musical rules. Finally, (Zhao et al., 2020) using an extensive knowledge of musical theory worked to create a lightweight variational auto-encoder model. While many papers have used musical theory extensively to improve music generation models, we hope to create a metric that will simplify the process of assessing these models numerically.

3 Motivation

Our group has an interest and background in music, so we wanted to incorporate this knowledge into our machine learning research. There are many models for music generation, but many popularly used metrics to judge them result in dramatically different qualities of musical pieces being considered relatively the same. Many music papers in artificial intelligence, to our knowledge, are often forced to resort to qualitatively comparing generated pieces when numerical metrics fail to show the contrast.

We want a metric that will more effectively ascertain how tasteful a composition sounds or perhaps how distasteful. Listening to the sample music generated by the relative self-attention transformer model inspired us to do something on this topic.

3.1 Definition of Serialism

The rules of serialism are as follows:

1. No note should be repeated until all 12 notes of a note row have been played.
2. The order of the row remains consistent throughout the piece.
3. Notes can be played at any octave.

3.2 Main Goal

We propose a metric based on the rules of serialism in western music theory, which we call the $JETSING$ metric. In western music, which most of us are accustomed to, there is a central tonality from which notes and keys are chosen to sound "good." Conversely, serialism has no central tonality. Instead, it is based on the concept of a "row." A row is a random ordering of the 12 unique half-step pitches in western music theory (C - B on the piano). In strict serialism, you may not repeat a pitch until all 12 pitches in a row have been played, the row must always appear in the same order, and notes are allowed to be in any octave. Because there is no central tonality in this method, music based on serialism tends to sound creepy, confusing, lost, and seemingly aimless.

Many models that audibly perform worse tend to have characteristics of serialism. They wander aimlessly, do not repeat motifs, and lack a sense of central tonality. Conversely, the models that perform well tend to have central tonality and can repeat motifs and create a long-term sense of direction in the piece.

Our $JETSING$ metric is based on the rules of serialism and captures how well a piece performs with regard to those rules. In other words, we measure how serial a piece is. The less serial a piece is, the better it adheres to western music theory and likely sounds "good" to our ears. $JETSING$ is comprised of two sub metrics $JETSING_{ROW}$ and $JETSING_{OCT}$. The former metric measures how well a piece adheres to the first two rules and the latter metric measures the average octave difference between each pair of melodic notes.

4 Method

4.1 Parsing MIDI Files

The goal of the MIDI parser is to obtain only the melody and disregard the rest. The parser is set up to process any MIDI file into a format with which our metric can easily make calculations. The parser uses the Python library music21 to output an object containing notes from the MIDI input. The object is a list of notes or chords depending on whether one note or several notes are played at the same time.

The parser is designed to first try and separate the melody and bass by detecting a midpoint in the music. The parser follows the C scale notation where the lowest note is C and the highest is B.

Note	Value
C	3
D	4
E	5
F	6
G	7
A	8
B	9

Table 1: The parser assigns values 3 - 9 to C - B.

These values are those used in calculating the mean of the notes in a MIDI file. As for octaves, the parser is using the octave information given by the music21 MIDI decoder. The octave range is from 1 to 10. The parser will total up the values for each note and octave separately and divide them with the total notes in the MIDI to get the middle note.

- $\mu_{note} = \frac{\sum note}{cardinality_{note}(MIDI)}$
- $\mu_{octave} = \frac{\sum octave}{cardinality_{octave}(MIDI)}$
- middle note $= \mu_{note} || \mu_{octave}$

With the middle note obtained, the parser now roughly knows whether a note is part of the bass or melody. If a note is higher than the middle note, then it is considered part of the melody. Otherwise the parser considers it as part of the bass.

4.2 $JETSING$

Based on the rules of serialism, we first create two sub-metrics of $JETSING$: $JETSING_{ROW}$ and $JETSING_{OCT}$.

4.2.1 $JETSING_{ROW}$

$JETSING_{ROW}$ will be responsible for the first two rules. It begins at 0 and increments for each break of rules 1 and 2. We divide this sum by the highest possible number of correct tone rows the piece could have had.

Here are examples and valid and invalid note orderings using just 5 tones:

- Valid Row: A-B-C-D-E
- Invalid (Violates 1): A-A-B-C-D-E
  - All notes are accounted for, but a note is repeated before all notes are played.
- Invalid (Violates 2): A-B-D-C-E

- All notes are accounted for without repetition, but the original order is not maintained.

We define JETSING-ROW below:

- $E = \sum errors$
- $M = (T/L)$
- $JETSING_{ROW} = (E/M)$

Where M represents the number of tone rows that fit within the given piece. (T representing the length of a tone row, and L representing the length of the piece). Calculating the $JETSING_{ROW}$ score this way gives an average over the whole piece with respect to the number of correct possible tone rows.

4.2.2 $JETSING_{OCT}$

This metric handles the third rule of serialism. Since the rule simply allows a pitch to be of any octave it doesn't have a binary rule break. In light of this we separated the rule into its own metric $JETSING_{OCT}$.

In this metric we measure the octave difference between notes p_t and p_{t+1}, where p is the pitch at time t. If these two notes are in the same octave the metric reports a 0. If the second note is at least one octave higher or lower the metric reports a 1 and so on. This metric reports the difference in octaves from one note to the next.

We define the $JETSING_{OCT}$ below:

- $\alpha_t = \Omega_{p_{t+1}} - \Omega_{p_t}$
- $A = \frac{\sum_{t=0}^{n-1} \alpha_t}{n-1}$
- $JETSING_{OCT} = \frac{A}{L-1}$

α_t represents the octave, Ω, difference of pitches p_{t+1} and p_t. We then sum over α at each time step, to get the total octave differences in the piece, and then divide by one less than the number of notes in the piece, n, to get the average octave difference over the piece, A.

For each octave difference between notes n_t and n_{t+1} the metric reports the octave difference. We take the sum of octaves differences and divide by number of pitches -1 in the piece.

Calculating the $JETSING_{OCT}$ this way yields the average octave difference from note to note over the whole piece.

4.2.3 $JETSING$

Now that we've defined the two submetrics to be used, we define the $JETSING$ metric to be:

$$JETSING = \frac{JETSING_{ROW}}{JETSING_{OCT}}$$

Higher $JETSING_{ROW}$ scores and lower $JETSING_{OCT}$ scores will make $JETSING$ as a whole get larger. Lower $JETSING_{ROW}$ scores and higher $JETSING_{OCT}$ scores will make $JETSING$ as a whole get smaller. A high $JETSING$ score indicates a good quality piece, while a lower score indicates it is more serial (and therefore less appealing).

5 Experiments

For this experiment we are using pretrained models to generate music, then rating their performances using our metric. The models we used were Music Transformer by (Huang et al., 2018), Performance RNN by (Simon and Oore, 2017), and MusicVAE by (Roberts et al., 2018).

As a baseline, we have applied JETSING to music created by humans and qualitatively analyzed its measurements. We then compare each model's average score on our metric to their scores with the metrics used by the original authors. Finally we compare our baseline results to the machine-generated results.

5.1 Datasets

Since we used pretrained models, we had no need to find data sets for this paper.

6 Results

7 Conclusion

8 Future Work

References

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.

Nikhil Kotecha and Paul Young. 2018. Generating music using an lstm network.

Adam Roberts, Jesse H. Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. 2018. A hierarchical latent vector model for learning long-term structure in music. *CoRR*, abs/1803.05428.

Ian Simon and Sageev Oore. 2017. Performance rnn: Generating music with expressive timing and dynamics. https://magenta.tensorflow.org/performance-rnn.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Yizhou Zhao, Liang Qiu, Wensi Ai, Feng Shi, and Song-Chun Zhu. 2020. Vertical-horizontal structured attention for generating music with chords.