RL-MUSICIAN: A TOOL FOR MUSIC COMPOSITION WITH REINFORCEMENT LEARNING

A DRAFT

Nikolay Lysenko
nikolay.lysenko.1992@gmail.com

January 26, 2020

ABSTRACT

In this paper, automatic composition of first species counterpoint is framed as an optimization problem close to reinforcement learning. An agent with shallow neural network as an actor model composes polyphonic pieces sequentially one measure (bar) per time step. Once a specified number of measures is added, resulting piece is evaluated based on some hand-written rules derived from music theory. Cross-entropy method is used for searching weights of actor model that result in maximum expected score of a piece.

Although current setup of the problem may seem simplistic, it is a step in a promising direction. Numerous recent advances in music generation are achieved with supervisedly trained neural networks and reinforcement learning is not used there as a standalone paradigm. Since collections of existing pieces are involved as datasets, results are "inspired" by these reference pieces. However, the notion of creativity is wider – it also includes generation of something that meets criteria of being an art, but is not backed by existing pieces. Thus, new aspects of creativity can be revealed by further research of pure reinforcement learning approaches.

Keywords: algorithmic composition · music generation · reinforcement learning · species counterpoint

1 Introduction

Algorithmic music composition is automatic generation of outputs representing musical pieces and written in some formal notation. To name a few of common notations, there is sheet music, tablature, and MIDI standard. It is not required from output representation to unambiguously define sound waveform. For example, sheet music leaves exact loudness of played notes to discretion of a performer and may include only imprecise hints like pianissimo (very quiet). Anyway, there are parameters of sounds that must be determined by their representation. Usually, such parameters are pitch, start time, and duration.

Currently, there are no perfect tools for algorithmic composition. Plenty of various approaches for composing music automatically exist, but none of them produce well-structured and novel pieces that cannot be distinguished from works of a talented human composer. Thus, there is an open research problem. Several recent breakthroughs in it are accomplished with machine learning and both supervised learning [1, 2] and reinforcement learning [3] are applicable. Moreover, some researchers combine them [4, 5, 6].

In this paper, a new approach to music composition is suggested. Although it is quite straightforward, to the best of the author's knowledge it is not described anywhere, so the first contribution is its rigorous definition. The second contribution[1] is proper setup and tuning of the parameters. It is often the case that accurately tuned simple methods outperform complex methods [7] and, in addition, they are more transparent and less demanding.

The approach relates to a well-known framework for simplified music composition called species counterpoint. It is often used by beginning composers in order to practice in voice leading (also known as part writing) and polyphony. From automatic composition point of view, species counterpoint is interesting due to two its properties:

[1] As of now, it is in progress.

- There are rules prescribing what to do and what not to do. So there are less degrees of freedom and the problem is computationally more feasible.
- There are clear hints of what is better to avoid and what should be added. So formal rules for automatic evaluation of resulting piece can be introduced.

The word "species" in the name of the framework indicates that there are some types – each of them has its own rhythmic pattern. Here, first species counterpoint is covered. This type has trivial rhythmic structure such that only whole (semibreve) notes can be used and every note must be placed in a beginning of a measure. Thus, interaction between rhythm and tonality is eliminated and the problem becomes even simpler.

Brief outline of the setup is as follows. There is an environment that keeps a piece in progress and the piece consists of a fixed number of measures. Each measure must be either empty (if it is not filled yet) or containing exactly one note per a melodic line. Initially, only the first measure and the last measure are filled with user-defined values. At a particular time step of an episode, an agent observes vector in which for every pitch there is an exponential moving average over all measures up to the current one (exclusively) of binary indicators whether this pitch is played at corresponding measure. An action of the agent is filling of the current measure, i.e., adding a pitch to each of the melodic lines. When the last measure is reached, episode is ended and the agent receives reward based on evaluation of created composition. Trainable parameters of the agent are weights of a neural network used as a so called actor model. These parameters are trained with cross-entropy method [8] (alternatively, they can be trained with genetic algorithms or evolutionary strategies [9]). From optimization perspective, average over several episodes reward is estimated value of fitness function at the weights used.

Obviously, the approach belongs to reinforcement learning, because there is an environment and an agent, but a dataset is absent. Nevertheless, there are some connections with supervised deep learning. From generation of new pieces point of view, actor model returns probabilities of next sonorities (chords) given current state. This is similar to sampling new sequences from a next-step prediction model trained with maximum likelihood method on a dataset of existing sequences.

More details on the methodology are provided in Section 3 but here it is appropriate to discuss its advantages. Of course, automatic composition of species counterpoint is not novel – some researchers have developed optimization-based approaches not only for first species, but also for fifth species [10]. Nevertheless, as far as the author knows, such approaches yield a single piece as a result of optimization, whereas in the current approach training produces an agent that creates a new piece after each run.

Next-step prediction models create a new piece after each run too, but they are trained supervisedly. The reasons for not involving supervised training at all are as follows:

- Finding new ways of music creation is a more challenging task than imitation of famous pieces. If no known pieces are used, chances are that the harder problem is considered and it is not replaced with the simpler problem of imitation.
- There are tuning systems other than equal temperament (for instance, in microtonal music). For some of them it may be impossible to collect dataset large enough to allow training models in a supervised fashion. However, developers of a tuning system should know some underlying principles and so (at least, in theory) it is possible to create evaluational rules and train an agent based on them.

Actually, results reported at this draft version are far from using above advantages at full scale (in particular, first species counterpoint is too limiting). The current study is rather a proof-of-concept, but this proof-of-concept is easily extendable and ideas on how it can be improved are listed in Section 5.

2 Background and Related Work

2.1 Algorithmic Composition

Usage of computers for automatic composition of music dates back to 1950s and, what is more, approaches that can be labeled as algorithmic are known since Guido of Arezzo (11th century). Therefore, it is impossible to list here all studies about the subject. An interested reader can find more information in specialized reviews such as [11]. What is discussed below is just some examples related to reinforcement learning.

Reinforcement learning can be applied to algorithmic composition either as a standalone paradigm or in connection with deep learning. As for the first case, probably, there are no influential works and only separate studies like [12] are available. Conversely, the second group recently gained attention and several interesting approaches were developed there.

For example, reinforcement learning can be used for altering weights of recurrent neural networks trained to generate music sequentially [4, 5]. The goal is to make generated pieces more structured and conformed with music theory rules. To define an environment, let its state be composed of recurrent neural networks states and previously played notes, let an action be an output for current time step and let reward depend on both evaluational rules and probability of output according to initial RNN. Rewards are granted immediately after a step and so DQN (Deep Q-Networks [13]) are preferred as a training algorithm.

Usually, softmax activation function is used in the last layer of generative RNN. However, it is a common practice in reinforcement learning domain to generalize it with Boltzmann softmax which is a family of activation functions parametrized by one parameter called temperature and denoted as t. Depending on temperature, output distribution can vary from atomic distribution concentrated at the most probable action ($t = 0$) to distribution returned with softmax activation ($t = 1$) and to uniform distribution ($t \to +\infty$). In [6], Boltzmann softmax is used and also input vector is extended by introducing an additional part (so called plan) indicating origination of initial input. An environment is defined so that state is a pair of temperature and plan, actions are changes in either temperature or plan, and reward depends on evaluation of produced with these settings piece.

Since algorithmic composition can be framed as training of a generative model, it sounds natural to try one of the most salient examples of generative models – generative adversarial networks (GANs) [14]. However, widespread notations for music assume sequences of discrete values but classical GANs work well only with continuous data, because gradient of discrete-valued functions is uninformative. Techniques originating from reinforcement learning can be used to overcome this obstacle [15, 16]. Namely, generator is trained with policy gradient method [17]. Such methodology is applied to various tasks and music composition is amongst them [18].

2.2 Cross-Entropy Method

Initially, cross-entropy method was developed for estimation of rare events probability [8]. However, it was found that it is also appropriate for solving optimization problems. More detailed discussion of cross-entropy method can be found in [19].

Algorithm 1 Cross-entropy method for optimization

Input: X – set of elements, $f : X \to \mathbb{R}$ – target function, $u(\cdot, w)$ – probabilistic distribution over X parametrized by vector w.
Output: $\hat{w}$ – approximate solution to the problem $\max_w \mathbb{E}_{x\sim u(\cdot,w)} f(x)$.
Hyperparameters: $w^{(0)}$ – initial value of w; N – number of iterations, n – number of vectors to draw at each iteration, σ – standard deviation for vectors generation, m – number of trials for each vector; ρ – fraction of best vectors to use for update; α – smoothing coefficient of updates.

for all $i \in \{1, \dots, N\}$ **do**
  for all $j \in \{1, \dots, n\}$ **do**
    draw $w^{(i,j)} \sim \mathcal{N}(\cdot|w^{(i-1)}, \sigma)$
    $r_j \leftarrow \sum_{k=1}^{m} f(x_k)$ where $x_k \sim u(\cdot, w^{(i,j)})$
  end for
  $r_{\text{threshold}} \leftarrow [\rho n]$-th highest value of $\{r_j : j \in \{1, \dots, n\}\}$
  $J \leftarrow \{j : r_j \geq r_{\text{threshold}}\}$
  $w^{(i)} \leftarrow \alpha w^{(i-1)} + (1 - \alpha)(\sum_{j\in J} w^{(i,j)})/[\rho n]$
end for
$\hat{w} \leftarrow w^{(N)}$

Algorithm 1 defines a variant of cross-entropy method for optimization. Sometimes, hyperparameter m is omitted and intermediate results are not aggregated over multiple trials. In case of $u(\cdot, w)$ that acts like a deterministic function of w, m is redundant, but, in general case, terminal result can be improved by setting $m > 1$.

In context of reinforcement learning, cross-entropy method is used for searching parameters of agent that result in maximum expected reward. So X is a set of all possible finished episodes, f is reward, and $u(\cdot, w)$ describes how an episode depends on parameters. Cross-entropy method is best suitable for environments where reward becomes known only after an episode is finished. If there are rewards granted after intermediate steps, methods based on Q-values usually outperform it.

3 Methodology

3.1 Setup

To start with, define representation of a musical piece. Suppose that counterpoint is written in l melodic lines (parts, voices), where l is usually 2 or 3. Suppose also that a diatonic scale (like C-major or A-minor) is chosen. Let each line be a list of pitches from the scale (i.e., only 7 notes per octave can be used and chromaticism is prohibited). Set length of all lines to the same number m, where m is usually between 8 and 24. From music point of view, this means that a piece to be created is rather a phrase that can be inserted into a longer composition. Such limitation goes back to species counterpoint, because it is designed exactly for writing short phrases. This way or that, define a piece as a matrix P of size $l \times m$ where element P_{ij} is a pitch played in i-th line in j-th measure.

Every line must have only small intervals between its successive elements in order to be perceived as a single line and not as a heterogeneous collection of sounds. Suppose that maximum allowed melodic interval is s scale degrees (here, $s = 3$). If so, there are no more than $(2s + 1)^l$ options to fill the next measure given current measure. Actually, the number is smaller, because some options are filtered out by counterpoint rules. The rules enforced in this study can be either rules of voice leading (part writing) or rules of harmony.

Here, five rules of voice leading are imposed:

- Only pitches from tonic triad (i.e., tonic, mediant, and dominant) can be rearticulated (repeated), because only they are stable;
- If a line has submediant followed by leading tone, tonic must be used after leading tone, because there is strong attraction to it; similarly, if a line has leading tone followed by submediant, dominant must be used after submediant;
- If a melodic interval between two successive pitches from the same line is larger than second (i.e., if there is a skip), the second pitch must be from tonic triad, because skip creates enough tension and something stable is needed;
- Movement in the opposite direction must happen immediately after a skip larger than third in order to resolve tension associated with the skip;
- For every pitch, there must be a way to reach the final pitch of this line with step motion (the final measure is filled initially with user-defined sonority), because resolution must be present during last measures.

In addition, there are two rules of harmony:

- All harmonic intervals between two simultaneously sounding pitches from different lines must be consonant;
- For every pitch, the closest to it pitch from the same measure must be no more than tenth apart.

Above rules are absolute and an agent cannot break them, because it selects pitches for the next measure only from options that are filtered according to them. If a set of such options is empty, episode terminates and an agent receives negative reward for dead end.

If the set of options is not empty, an agent needs an observation. Observation is provided in terms of so called piano roll format. Denote size of pitch range available to the agent as n and let the piece be represented also as a matrix R of size $n \times m$ where R_{ij} is 1 if i-th pitch is played in any line in j-th measure and 0 otherwise. Denote current measure as t, $1 \leq t \leq m$. Then observation is defined as vector v of length n:

$$v = \sum_{i=1}^{t} \beta^{t-i} R_{\cdot i},$$

where β is a hyperparameter related to exponential decay, $0 < \beta < 1$.

For each option, agent concatenates observation v with encoding this option vector u of length $(2s + 1)l$. Exactly one element is 1 and others are 0 amongst the first $(2s + 1)$ elements of u; position of the non-zero element corresponds to movement of the first line; the same encoding procedure is valid for other blocks of size $(2s+1)$ and lines corresponding to them. Concatenation of v and u is passed as input to so called actor model which is a shallow neural network returning a single real-valued score. After these scores are collected for all valid options, Boltzmann softmax is applied to them in order to convert them to probabilities of actions. An action is drawn from this distribution and the agent takes it.

Remember that an episode starts with the first and last measures already filled. The agent fills empty measures one-by-one according to the described above procedure. When the final measure is reached, the episode ends, created piece is evaluated and the agent receives reward.

3.2 Evaluational Rules

At the current revision of the study, eight properties of a final piece are evaluated separately and total reward is a weighted sum of these eight scores.

- **Autocorrelation.** Pieces where the same sequence of sounds is repeated over and over again are worse than pieces with no repetitions. To reflect this in reward, this score is used:
$$1 - \max_k \{|\mathrm{Autocorr}_k(R)| : 2 \leq k \leq 8\},$$
where R is piano roll representation of the piece, $\mathrm{Autocorr}_k$ is autocorrelation function for lag k (for matrix R its autocorrelation is average autocorrelation of its rows) and maximum lag to consider is set to 8 based on common sense.
- **Entropy of pitch distribution.** To encourage usage of all pitches from a range corresponding to a line, there is a score equal to average over all lines actual entropy of pitch distribution within the line normalized by entropy of uniform distribution on the same set of pitches.
- **Absence of looped pitches.** A stable pitch may be rearticulated, but not more than once in a row. Every excessive occurrence of the same pitch within a line adds constant negative reward.
- **Absence of pitch or pitch class clashes.** According to species counterpoint methodology, lines must be as distinguishable from each other as possible. In particular, this implies that the same pitches or the same pitch classes should not sound simultaneously in different lines. What is more, unison or octave harmonic intervals are too stable and so they create illusion of false finish if they occur in intermediate measures. To take into account these considerations, there is a negative reward proportional to share of intervals with both pitches belonging to the same pitch class amongst all intervals between two simultaneously sounding pitches. Also there is extra negative reward proportional to share of unison intervals.
- **Motion.** For every pair of lines every motion between two successive measures adds a reward that depends on a type of motion. Since contrary motion makes lines more distinguishable, reward for it is positive. Reward for oblique motion is zero, reward for similar motion is slightly negative and reward for parallel motion is negative, because parallel motion makes lines sounding like copies of each other.
- **Correlation between lines.** Consider each line as a sequence of real numbers equal to pitch positions within a range of available pitches. Then half of difference of 1 and average over all pairs of lines correlation between two lines is returned as a reward for independence of lines.
- **Explicity of climax.** Music theory suggests that good melodies must have goal-oriented motion and that such goal-orientedness may be achieved by having exactly one climax point. For each line, negative rewards are granted for each duplication of climax point and for not so high climax point.
- **Number of skips.** Positive reward is associated with every line such that number of skips in it lies within a specified range. The motivation behind this is that lines with almost no skips are not interesting, but lines with too many skips are not coherent.

It can be seen that the first three properties deal with non-triviality of results, the next three properties deal with interaction of lines and the last two properties deal with melodic quality of each separate line.

4 Experimental Results

A software implementation of the above methodology in Python programming language is available on GitHub[2]. The code has built-in documentation, is covered with unit tests, and is released as a package on PyPI[3]. All important settings are placed to a separate configuration file and so it is easy to experiment with them.

The implementation relies on some open-source tools [20, 21, 22, 23].

In this section, results are reported for experiments with 2 melodic lines and 16 measures starting from (G4, C5) and ending with (C4, C5). It is found that scores for autocorrelation and entropy of pitch distribution are redundant. Non-degenerate pieces are produced even without them. So only six properties are evaluated. They are weighted so that explicity of climax and absence of pitch class clashes have 3 times more impact than any other property.

In Table 1, scores of a typical piece created by an agent trained for 20 populations, are compared with two benchmarks. The first one is simple baseline – it is score of a piece created by an agent with weights that are the best amongst 100 random weights. The second one are scores for a piece manually created by the author in about 15 minutes.

[2] `https://github.com/Nikolay-Lysenko/rl-musician`

[3] `https://pypi.org/project/rl-musician/0.2.0/`

Table 1: Scores of sample pieces

	Baseline	Agent	Human	Maximum
Absence of looped pitches	-1	0	0	0
Absence of pitch or pitch class clashes	-0.429	-1.286	-0.643	0
Motion	-0.1	-0.033	0.1	1
Correlation between lines	0.552	0.168	0.278	1
Explicity of climax	0.9	3	3	3
Number of skips	0	1	1	1
Total reward	-0.076	2.849	3.735	6

It can be seen that the agent's piece consists of two lines with formally good melodies but these lines are not independent enough. Table 2 contains the created by the agent piece for which above scores are reported. Visual analysis of the shown piano roll confirms observation about lines dependence – their climax points coincide and there are plenty of intermediate octave intervals.

Table 2: A piece composed by an agent

							C6								
						B5									
					A5				A5						
				G5				G5							
			F5								F5				
		E5								E5					
	D5												D5		
C5							C5					C5		C5	C5
								B4							
									A4						
G4				G4		G4				G4					
	F4				F4						F4				
												E4			
			D4										D4		
		C4												C4	C4

As for subjective evaluation of music quality, it is not correct to compare the selected piece created by the agent with the piece created by the author. The problem is that reward encourages to follow counterpointal rules but, from a modern listener point of view, breaking these rules does not necessarily sound unaesthetically. That being said, subjective perception of a resulting piece is not aligned with fitness function used during optimization. However, it must be noted that the piece shown above looks like excessively oriented on maximization of fitness function regardless more subtle properties important for human listeners. It is not a problem associated with training, it is a flaw in problem setup (more criteria for evaluation are needed).

Numerous experiments revealed another important issue. Actually, different runs of a trained agent produce variations of the same piece. Although there might be significant differences in scores (in particular, because some scoring functions are discrete), there are no significant differences from music point of view.

5 Conclusion

This paper studies capabilities of reinforcement learning for music composition without usage of existing pieces. Here, music composition is framed as optimization problem where the goal is to find values of generative model's parameters maximizing expected score of pieces generated with them. It is found that some progress can be made in this direction.

The list of suggested further improvements is as follows:

- **Reflect more knowledge from music theory in filtering rules and evaluational rules.** For example, overlapping motion should be discouraged. Also downward skips can be penalized in order to have upward skips before climax and downward conjunct motion after it.
- **Promote variety in fitness function.** Now, fitness function just averages rewards for several episodes. However, a new term can be added to this average. This term must reflect how different created pieces are from each other.
- **Revise feature representation for actor model.** Probably, procedure of next measure generation should be changed. Currently, it is assumed that shallow neural network is able to act based on relatively raw feature representation. To make it less raw, some features inspired by music theory may be engineered.
- **Tune hyperparameters.** In the current software implementation, cross-entropy method has five hyperparameters, an agent has two hyperparameters, and evaluation functions have more than ten hyperparameters. As of now, not so many computational experiments are held in order to find their optimum values.
- **Abstract away all rules and focus only on essentials of music.** Tonal music introduces attraction between pitches. Depending on this attraction and degree of consonance/dissonance, tension increases or dissipates. Moments of time when level of tension changes may be rhythmically accented or unaccented. Close but not exact alignment between rhythmical accent and resolution often results in aesthetic and interesting music. Taking into account constraints of smooth voice leading, optimization problem can be set up based on above considerations.

References

[1] Daniel Johnson. Generating Polyphonic Music Using Tied Parallel Networks. In *International Conference on Evolutionary and Biologically Inspired Music and Art*, pages 128–143, 03 2017.

[2] Christine Payne. MuseNet. `openai.com/blog/musenet`, Apr 2019.

[3] B.D. Smith and G.E. Garnett. Reinforcement Learning and the Creative, Automated Music Improviser. *Evolutionary and Biologically Inspired Music, Sound, Art and Design. EvoMUSART. Lecture Notes in Computer Science*, 7247, 2012.

[4] Natasha Jaques, Shixiang Gu, Richard E. Turner, and Douglas Eck. Generating Music by Fine-Tuning Recurrent Neural Networks with Reinforcement Learning. 2016.

[5] Nikhil Kotecha. Bach2Bach: Generating Music Using a Deep Reinforcement Learning Approach. *arXiv e-prints*, page arXiv:1812.01060, Dec 2018.

[6] Harish Kumar and Balaraman Ravindran. Polyphonic Music Composition with LSTM Neural Networks and Reinforcement Learning. *arXiv e-prints*, page arXiv:1902.01973, Feb 2019.

[7] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. Are We Really Making Much Progress? A Worrying Analysis of Recent Neural Recommendation Approaches. In *Proceedings of the 13th ACM Conference on Recommender Systems*, pages 101–109, 2019.

[8] Reuven Y. Rubinstein. Optimization of Computer Simulation Models with Rare Events. *European Journal of Operational Research*, 99(1):89–112, 1997.

[9] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution Strategies as a Scalable Alternative to Reinforcement Learning. *arXiv e-prints*, page arXiv:1703.03864, Mar 2017.

[10] Dorien Herremans and Kenneth Sörensen. Composing Fifth Species Counterpoint Music with Variable Neighborhood Search. 2012.

[11] Jose David Fernandez and Francisco Vico. AI Methods in Algorithmic Composition: A Comprehensive Survey. *Journal of Artificial Intelligence Research*, 48:13–582, Feb 2013.

[12] Liangrong Yi and Judy Goldsmith. Automatic Generation of Four-Part Harmony. In *Fifth UAI Bayesian Modeling Applications Workshop*, Jan 2007.

[13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning. *arXiv e-prints*, page arXiv:1312.5602, Dec 2013.

[14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27*, pages 2672–2680. 2014.

[15] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. *arXiv e-prints*, page arXiv:1609.05473, Sep 2016.

[16] R Devon Hjelm, Athul Paul Jacob, Tong Che, Adam Trischler, Kyunghyun Cho, and Yoshua Bengio. Boundary-Seeking Generative Adversarial Networks. *arXiv e-prints*, page arXiv:1702.08431, Feb 2017.

[17] Ronald J. Williams. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Mach. Learn.*, 8(3-4):229–256, May 1992.

[18] Gabriel Lima Guimaraes, Benjamin Sanchez-Lengeling, Carlos Outeiral, Pedro Luis Cunha Farias, and Alán Aspuru-Guzik. Objective-Reinforced Generative Adversarial Networks (ORGAN) for Sequence Generation Models. *arXiv e-prints*, page arXiv:1705.10843, May 2017.

[19] P. T. de Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein. A Tutorial on the Cross-Entropy Method. *Annals of Operations Research*, 134(1):19–67, 2005.

[20] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv e-prints*, page arXiv:1606.01540, Jun 2016.

[21] François Chollet et al. Keras. `https://keras.io`, 2015.

[22] Travis E. Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

[23] Colin Raffel and Daniel P. W. Ellis. Intuitive Analysis, Creation and Manipulation of MIDI Data with pretty_midi. In *Music Information Retrieval Late Breaking and Demo Papers, 15th International Conference on*, 2014.