# RL-Musician: A Configurable Open-Source Tool for Species Counterpoint Composition

A draft

**Nikolay Lysenko**
nikolay.lysenko.1992@gmail.com

May 16, 2020

**Abstract**

In this paper, automatic composition of fifth species counterpoint is framed as a tree search problem close to reinforcement learning. The goal in this problem is to compose a melodic line given another melodic line consisting of whole notes only. The line in progress has fixed ends and the gap between these ends is filled sequentially from start to finish. Once it is done, an episode terminates and the resulting piece is evaluated based on some hand-written rules derived from music theory. Best sequences of notes are found with a variation of Monte Carlo Beam Search.

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 to 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, 4] are applicable. Moreover, some researchers combine them [5, 6, 7]. Meanwhile, fitness function optimization also yields prominent results[8, 9].

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 [10] 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 usually 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, fifth species counterpoint is covered. This type is the highest one and the less restrictive one. Like in other species, there is a given melodic line consisting entirely of whole (semibreve) notes. This line is called cantus firmus. Again, like in other species, a new melodic line of the same duration is going to be created (so called counterpoint line) and these two lines should sound aesthetically pleasant when played simultaneously. The difference between fifth species and other species is that a counterpoint line may change its rhythmic pattern: half notes, quarter notes, eighths notes, and suspended whole notes can be used.

Brief outline of the setup is as follows. There is an environment that keeps a piece in progress. Initially, the piece consists of a user-defined cantus firmus, the first note of counterpoint line, and the last note of counterpoint line. An action is extension of the counterpoint left part by one new note. Not every action is valid, there are rules prohibiting some of them. Invalid actions can not be done by design. When the last note is reached, episode is ended and a reward based on evaluation of the created composition is assigned to the sequence of passed actions.

The remainder of the paper is organized as follows. Section 2 describes the most relevant works and the place of the current paper amongst them. Namely, Section 2.1 discusses algorithmic composition, whereas Section 2.2 contains preliminaries on Monte Carlo Beam Search and lists modifications made in this study. Details of the setup (in particular, rules used for evaluation of pieces) are defined in Section 3. Section 4 reports experimental results and the way to reproduce them. Finally, Section 5 concludes.

## 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 the current study.

These examples can be divided into two groups. The first one is formed by approaches where an explicitly given fitness function is optimized. The second group is formed by composition of music with reinforcement learning in its narrow sense (i.e., methods based on value function estimation). This group is something in between the first group and supervised learning approaches. On the one hand, reward function is a domain-specific name for fitness function. On the other hand, data generated during past episodes are used for training supervised models that approximate value function.

The closest to the current study works [8, 9] belong to the first group. A local search algorithm named Variable Neighborhood Search (VNS) is used there in order to create first and fifth species counterpoint pieces that maximize fitness function. What these studies and the current one share in common is that fitness (reward) function is backed by the theory of species counterpoint. Despite that, there are important differences:

- In [8, 9], the whole piece is created at random and then improved by applying small alterations. As a result, there are only evaluation rules and there are no rules of composition. If a piece breaks some fundamental rules of counterpoint, it is still evaluated (but it can receive score making it unfeasible). Conversely, in the current study, both rules of composition and rules of evaluation are present as separate entities. An action that contradicts a rule of composition can not be done and so only compliant pieces are evaluated.
- In [8, 9], evaluation rules accurately follow counterpoint theory, but here it is not reproduced too literally. From a modern listener point of view, violation of counterpoint prescriptions does not necessarily sounds unpleasant, because some of these prescriptions are style-specific.
- Output of VNS is a fixed piece. If cantus firmus is changed, VNS should be run from scratch. Actually, output of Monte Carlo Beam Search is a fixed piece too and search should be run from scratch for every new cantus firmus as well. However, the current approach can be upgraded to a true reinforcement learning approach. Such upgrade is possible, because the problem is formulated in RL terms: composition of a piece is considered an episode consisting of sequential steps after each of which a list of valid actions updates. If so, there can be an agent that observes current state of a piece and writes counterpoint line based on observations. Cantus firmus can be changed

from episode to episode during training of the agent in order to teach the agent to deal with various cantus firmi. After each episode played by the trained agent, a new output piece can be created even for a fixed cantus firmus.

- Methodology of the two discussed papers has a software implementation named Optimuse, but download links are broken and so results can not be reproduced. Unlike this, links from the present paper lead to time-tested sites such as GitHub and PyPI.

There are other optimization-based approaches as well. Instead of local search metaheuristics, population-based metaheuristics (genetic algorithms, evolutionary strategies, cross-entropy method), or construction metaheuristics (ant colony algorithms) can be used. However, such papers are less relevant to the current study and so let us switch our attention to reinforcement learning. It can be applied to algorithmic composition either as a standalone paradigm or in connection with supervised deep learning. As for the first case, some separate studies like [3, 4] are available, but 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 (RNNs) trained to generate music sequentially [5, 6]. 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 DQN (Deep Q-Networks [12]) 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 by $t$. 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 [7], 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) [13]. 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 [14, 15]. Namely, generator is trained with policy gradient method [16]. Such methodology is applied to various tasks and music composition is amongst them [17].

Let us highlight again that above cases of reinforcement learning usage still require supervised learning and datasets. The arguments for not involving them 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 a 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 define a fitness function and find pieces that optimize it or to define a reward and train an agent based on it.

This implies that optimization-based approaches and pure reinforcement learning approaches must not be overlooked due to advances in supervised learning application.

### 2.2 Monte Carlo Beam Search

Consider a tree search problem. To be more detailed, suppose that there is a tree such that every its leaf is mapped to a reward and the goal is to find the leaf with the maximum reward. However, leaves and corresponding to them rewards are not known initially. Information about a leaf becomes available only after a path from the root of the tree to this leaf is constructed.

The described in Section 1 procedure of fifth species counterpoint writing can be considered a tree search problem. The root is the initial state of an environment. Each valid option to extend counterpoint line is an edge from the current node of the tree to a new node. Leaves are pieces where counterpoint line is finished.

If size of a tree is small enough, exhaustive search is possible. There are well-known classical algorithms such as depth-first search (DFS) that allow iterating over all paths from the root to leaves. Unfortunately, trees are large in numerous real-world problems.

An obvious idea for a large tree is to use random search. To create a path, every next node is chosen at random until a leaf is reached. After specified number of paths are created, the best one amongst them is chosen as an approximate solution.

This idea can be further improved in the following way. Let an optimum path be constructed sequentially with adding one new node per a series of random trials. If so, there is a stub initially containing only the root. At every iteration, there are several trials to continue stub at random until a leaf is reached. The best one amongst them (and amongst trials from previous iterations if applicable) is chosen and the first non-stub node from there is added to the stub. This method can be called Monte Carlo Search. The gain in comparison with plain random search is due to allocating more exploration to subtrees that contain the current best leaf.

Above version of Monte Carlo Search is greedy, because it ignores that some subtrees might be underexplored. A well-known method that explores more than greedy search but uses less computational resources than exhaustive search, is called beam search. Instead of just one stub, $w$ distinct stubs are constructed and used, where $w$ is a hyperparameter named beam width. It is natural to call this method Monte Carlo Beam Search.

Probably, the term Monte Carlo Beam Search was coined in [18]. However, it has different meaning there. In [19], the same author defined Nested Monte Carlo Search as a combination of Monte Carlo Search with DFS. Given a current stub, all its extensions by $n$ new nodes (where $n$ is a hyperparameter named level of nestedness) are found with DFS and then continued at random. So pre-defined number of trials per iteration is absent, because number of trials is equal to number of extensions by $n$ nodes ahead. Comparing with Monte Carlo Search, Nested Monte Carlo Search distributes exploration more evenly between short extensions of a stub. This way or that, in [18], Monte Carlo Beam Search is defined as beam search based on Nested Monte Carlo Search.

Nevertheless, here, beam search is based on Monte Carlo Search. The reason for it is quite mundane – parallel implementation of DFS looks like overcomplication that may yield too few. The second difference with [18] is that dependence of trials number on stub length is introduced. The longer the stub is, the smaller a subtree to be explored is. Therefore, it is better to reallocate computational resources in order to explore more when a stub is short and to explore less when a stub is long.

## 3 Methodology

### 3.1 Setup

To start with, define representation of a musical piece. Suppose that a diatonic scale (like C-major or A-minor) is chosen. Let a line be a list of $(\mathrm{pitch}, \mathrm{duration})$ pairs where $\mathrm{pitch}$ must be from the scale (i.e., only 7 out of 12 notes per octave can be used and so chromaticism is prohibited) and $\mathrm{duration} \in \{1, 2, 4, 8\}$ (i.e., it is measured in eighths of measure). In this study, pieces are written exactly in 2 melodic lines (parts, voices) where one of them is cantus firmus and the other one is counterpoint. Definition of cantus firmus implies that every its element has duration equal to 8 and duration of a piece in measures is equal to length of cantus firmus. Denote this length by $m$, 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.

Alternatively, a piece can be represented by a piano roll which is a matrix $P$ of size $n \times 8m$ where $n$ is size of available pitches range (hence, $n \leq 88$, since there are 88 piano keys) and $8m$ is number of eighths of measure in a piece lasting $m$ measures. Matrix element $P_{ij}$ is 1 if $i$-th pitch is sounding at $j$-th eighth and 0 otherwise.

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, $s = 7$). If so, there are no more than $(2s + 1)$ options to select the next pitch given the current pitch and there are no more than $4(2s + 1)$ options to add the next note given the current note (where $4 = \#\{1, 2, 4, 8\}$). Actually, the number is smaller, because some options are filtered out by counterpoint rules. The rules enforced in this study can be rules of rhythm, rules of voice leading, and rules of harmony.

Here, only one rule of rhythm is present:

- There must be no conflicts between strong beats and longer notes. In other words, every measure must be split in a manner that allows a listener to unambiguously define meter. For example, if a measure is split like this: (4, 2, 1, 1) or (2, 1, 1, 8) where 8 stands for a tied over bar note, this is correct, but if a measure is split like this: (2, 4, 2), this is not correct, because downbeat note is shorter than the next note and intermediate strong beat is inside a span of a note, not at a beginning of a note.

Further, seven 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;
- No pitches can be rearticulated two times in a row, because it makes a line stalling;
- A line can not move from an intermediate pitch by more than 9 semitones without any changes in direction (rearticulation is not a change of direction);
- 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;
- There can be no more than 2 skips (leaps) in a row;
- Movement by step in the opposite direction must happen immediately after a skip larger than third in order to resolve tension associated with the skip;
- For every note from counterpoint line, there must be a way to reach the final note of this line with step motion (remember that the final note is set initially), because resolution must be present during last measures.

Finally, there are seven rules of harmony:

- If a note from a counterpoint line starts on a strong beat (downbeat or measure middle), harmonic interval between the pitch and simultaneously sounding pitch from cantus firmus, must be consonant (although, if counterpoint note is suspended, it can be dissonant with the second simultaneously sounding note from cantus firmus);
- If a note from a counterpoint line forms a dissonance with cantus firmus, it must be reached by step motion;
- If a note from a counterpoint line forms a dissonance with cantus firmus, it must be left by step motion;
- If a dissonance is created by suspended note, this dissonance must be resolved by downward step motion;
- Harmonic intervals larger than tenth are prohibited;
- Voice crossing is prohibited (i.e., counterpoint can be either strictly above cantus firmus or strictly below it);
- Overlapping motion is prohibited (i.e., counterpoint can be either strictly above lagged cantus firmus or strictly below it).

Above rules are absolute and it is not possible to break any of them, because line continuations can be selected only from options that are filtered according to these rules. If a set of such options is empty, episode terminates and an unfinished sequence of actions receives negative reward for dead end.

It worths to be mentioned that the listed rules are based on counterpoint theory, but they do not reproduce it exactly. For example, there is a pattern named Double Neighbor and it is allowed in counterpoint music, but it is prohibited by the rule about step motion to a dissonating element.

Initially, cantus firmus, start note of counterpoint line (occupying the whole first measure except opening pause), end note of counterpoint line (occupying the whole last measure), and range of pitches available for usage in counterpoint line, are set externally. These values are the same for all episodes of Monte Carlo Beam Search. During each episode, a gap between the first note and the last note is reduced by one new note added to the left and this continues until the counterpoint line is finished.

## 3.2 Evaluational Rules

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

- **Absence of looped fragments.** Define a fragment as content of four or more consecutive columns of a piano roll $P$. For every fragment followed by exactly the same fragment, a constant negative reward is added.
- **Entropy of pitch distribution.** Let $C$ be a set of all pitches available for usage in a counterpoint line and let $\mu$ be the probability distribution on this set $C$ derived from frequencies of pitches in a counterpoint line. Divide entropy of $\mu$ by entropy of the uniform distribution on $C$. The result is a real number from 0 to 1. This number is returned as a reward for non-degeneracy of pitch usage.

- **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 a counterpoint line, a constant reward is decreased for each duplication of climax point and for not so high climax point.
- **Absence of aimless fluctuations.** If melody fluctuates within a narrow range of pitches for a long time, it sounds boring. So for every 9 or more successive notes from a counterpoint line such that the lowest of them and the highest of them are no more than third apart, constant negative reward is assigned. Also, if an interval between the lowest of 9 or more successive notes and the highest of them is a fourth, half of this negative reward is assigned.
- **Number of skips.** Depending on number of skips in counterpoint line, non-negative reward is granted. The motivation behind this is that lines with almost no skips are not interesting, but lines with too many skips are not coherent. So the reward is 0 when number of skips is 0, then it increases, reaches maximum at 3 skips and decreases after that becoming 0 again for 7 or more skips.

It can be seen that the first two properties deal with non-triviality of results, while the other three properties deal with melodic quality of a counterpoint 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].

In this section, results are reported for experiments with a cantus firmus attributed to Johann Joseph Fux, a theorist of species counterpoint. This cantus firmus lasts 12 measures. Counterpoint line starts with G4, ends with C5, and is allowed to have pitches from C4 to B5. Properties to be evaluated have equal weights.

Below figure demonstrates a piece produced with this setup. As Table 1 shows, the piece is close to theoretical optimum.

Figure 1: An example of a generated piece.

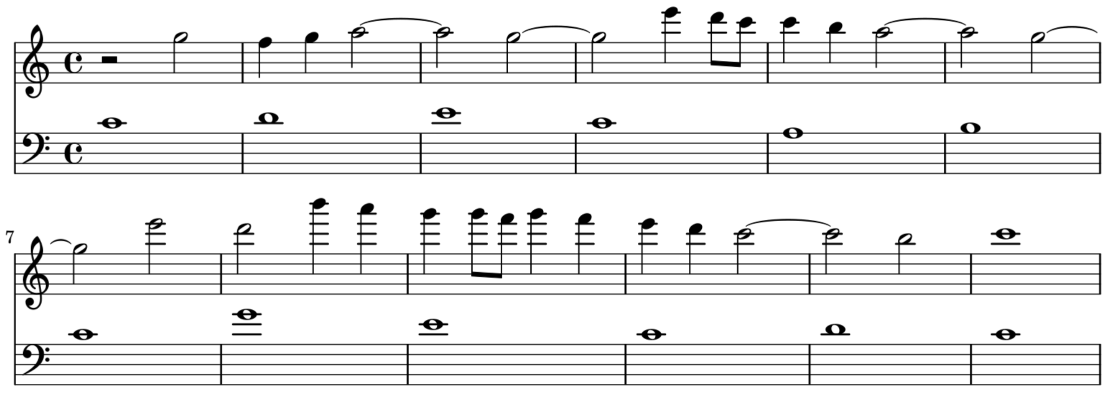

Table 1: Scores of the above piece.

| | Score | Maximum |
|---|---|---|
| Absence of looped fragments | 0 | 0 |
| Entropy of pitch distribution | 0.868 | 1 |
| Explicity of climax | 1 | 1 |
| Absence of aimless fluctuations | 0 | 0 |
| Number of skips | 1 | 1 |
| **Total reward** | 2.868 | 3 |

[2]`https://github.com/Nikolay-Lysenko/rl-musician`
[3]`https://pypi.org/project/rl-musician/0.4.1/`

The piece was found with beam width set to 5 and number of trials schedule set to $(30000, 10000 \times 5, 5000 \times 3, 1000 \times 5, 100 \times \ldots)$ – this means that the second note of the counterpoint line was chosen after 30000 random continuations and, for example, the eighth note of the counterpoint line was chosen after 5000 random continuations.

Multiple runs of Monte Carlo Beam Search were held and almost all of them led to scores similar to the reported above scores. Pieces found by different runs are not variations of each other, but they may share something in common. Taking into account that cantus firmus is the same for all of them, this is expected. Moreover, for the chosen cantus firmus and the chosen settings, climax pitch of a counterpoint line can be B5 (the highest allowed pitch) only in the eighth measure and this limitation obviously reduces variability.

## 5 Conclusion

This paper introduces a new approach to composition of music without usage of existing pieces. For a problem of fifth species counterpoint writing, the approach finds pieces that maximize weighted sum of associated with particular musical properties scores subject to boundary constraints. These boundary constraints (the first note and the last note) are user-defined and weights for the sum of scores are user-defined too.

Because the problem is a reduced version of general music composition problem, results can be virtually perfect from formal metrics point of view. However, several aspects of real music are ignored here:

- **Long-term structure.** The developed method produces short phrases (up to a pair of dozens of measures). When creating a longer piece, new concepts arise: say, motifs.
- **Higher number of voices.** In most Western pieces, four voices are used. Usually, one of them forms main melody and the others form accompaniment.
- **Freer textures and rhythmical patterns.** Presence of a melody entirely consisting of whole notes is not a mandatory requirement. Further, pauses can be used within lines. Finally, the higher number of voices is, the more rhythmical patterns become available.

It is impossible to take these aspects into account while staying within species counterpoint. A framework going beyond it should be involved.

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

[4] 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.

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

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

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

[8] Dorien Herremans and Kenneth Sörensen. Composing First Species Counterpoint with a Variable Neighbourhood Search Algorithm. *Journal of Mathematics and the Arts*, 6(4):169–189, 2012.

[9] Dorien Herremans and Kenneth Sörensen. Composing Fifth Species Counterpoint Music with Variable Neighborhood Search. *Expert Systems with Applications*, 2013.

[10] 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.

[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] 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.

[13] 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.

[14] 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.

[15] 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.

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

[17] 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.

[18] Tristan Cazenave. Monte Carlo Beam Search. *IEEE Transactions on Computational Intelligence and Ai in Games*, 4:68–72, Mar 2012.

[19] Tristan Cazenave. Nested Monte Carlo Search. *IJCAI International Joint Conference on Artificial Intelligence*, pages 456–461, Jan 2009.

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

[22] 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.