Music Generation Using Q-Learning

Mint Lin
School of Engineering
University of Virginia
Charlottesville, Virginia
xl9yr@virginia.edu

Yuchen Sun
College of Arts and Sciences
University of Virginia
Charlottesville, Virginia
ys4aj@virginia.edu

Simon Zhu
McIntire School of Commerce
University of Virginia
Charlottesville, Virginia
mz4cr@virginia.edu

***Abstract*—Music generation is an emerging field in artificial intelligence. It allows musicians and music lovers to compose music when they lack inspirations as well as to enjoy music in the style they desire. Existing projects like Magenta have successfully generated polyphony based on a given monophony. However, none has particularly focused on a two-line polyphony creation, or duet creation. Since the duet composition is a very common and useful practice for musicians, we intend to create an Artificial Intelligence that can transform one line of monophonic melody into a two-line polyphonic melody using Magenta's Polyphony_RNN_Generator and Q-learning. It takes in one line of music without accompanying chords, creates another monophonic melody that harmonizes with the input melody, and eventually outputs the resulting combined polyphonic melody. By doing this, we aim to provide one elementary solution to the problem of automatic music creation.**

***Index Terms*—music generation, artificial intelligence, Q-learning, transfer learning**

I. Introduction

AI application in generating polyphony is important because it offers composers and music lovers a shortcut for producing music. It is expected that the composers or producers require months or sometimes even years to accomplish a masterpiece under usual circumstances. A significant amount of time must be contributed to making polyphony, or harmonic melody, to add texture to the piece and enrich the music's quality. Polyphony AI shortens the time commitment for this process tremendously so that anytime composers with a new inspiration can produce an original masterpiece within just weeks instead of months.

Currently, the field of AI-generated music has attracted numerous attention from researchers. For instance, in its Magenta project, an open-source research project that specifically targets music creation with machine learning tools, Google has recently introduced Music Transformer, a self-attention-based neural network that can generate music with long-term coherence [1]. It has also introduced numerous other models such as Coconet and PerformanceRNN to facilitate music generation [2]. These models, particularly Polyphony_RNN_Generator (we will refer to this module as PolyRNN in this paper), will serve as the basis of our model [3].

On a higher philosophical level, it is also interesting to discuss whether machines or technologies can produce music that is equally beautiful or at least as acceptable as those made by humans. People have now seen the application of AI on many occasions, yet such events often require no more than constant repetition or massive computation. In other words, seldom has AI been applied to an area like arts or music where even people's judgment often vacillates and is constantly disparate. In these situations, without a unified standard specifying what kind of product is acceptable, and what type of work is impeccable, it remains a question whether AI could generate a piece that is at least accepted by many, if not purely appreciated by the whole society.

The provocative nature of this subject simultaneously explains its novelty. If the previous question leads to a reasonably positive result, people will embrace automation in another industry. Using such progress as a stepping stone, one can imagine that soon AI can be further developed to apply in other previously untouchable areas.

II. Methodology

A. Preprocessing Pipeline

Our project starts by selecting viable datasets. By investigating several options including Flute Dataset, Groove MIDI Drumming dataset, MAESTRO, sheet music from MuseScore.com, and Bach-Doodle dataset, we discovered similar potential issues for these datasets: either some music pieces are composed of polyphonic melodies instead of monophonic melodies or the music pieces contained in the dataset cannot be downloaded in batch [4]- [7]. Limited by such conditions, we finally arrives at the Bach-Doodle dataset, which contains 21.6 million distinct monophonic music entries that can be downloaded in batch. Each music entry from the Bach-Doodle dataset represents one monophonic melody. With all the Bach-Doodle melodies as input, we then utilized Magenta's PolyRNN, which generates a number (10 in this project) of possible polyphonic music pieces from the single-line melodies given. After processing the polyphonic music, root notes from the original Bach-Doodle single line melody are separated from all the new notes generated by PolyRNN. With data of all new notes and original root notes, we would be able to establish Q-learning States according to the extra information (e.g. note start time and end time) contained in the data structure of the notes. Also, we would be able to create a Q-learning Agents, having it parsing through the Q-learning States we established so that it can select the highest rewarded notes to compose a new single-line melody, which would then be added to the original melody to produce the duet. We also

desire to implement transfer learning to further optimize our result, so that multiple music files from the dataset can be utilized to fine-tune the Q-learning process.

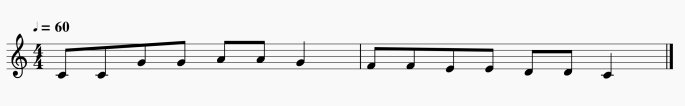

Fig. 1. Initial Monophonic Melody

B. Q-learning Agent

The Q-learning agent we implement works based on sample-based Q-value iteration. The formula is as follows.

$$Q_{k+1}(s,a) \leftarrow \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma \max_{a'} Q_k(s',a')]$$

In this formula, s refers to the current State, s' refers to the next State, a refers to the current chosen action, or in our case, the note chosen, and a' refers to the action or note chosen in the next State. R refers to the reward function, which will be defined later in the paper, and γ refers to the discount factor. In our case, $\gamma = 1$ as time of the reward is not a major concern in music creation. In our case, the environment is deterministic, considering that the agent will move to the exact note after it makes a decision. Therefore, the Q-value for a particular combination of (s, a) at $k + 1$ iteration only depends on one s'.

For each iteration, the agent will receive a sample (s, a, s', r). Based on the sample, it will yield a sample Q-value, which is

$$sample = R(s,a,s') + \gamma \max_{a'} Q(s',a').$$

Then, it will combine the old estimate of Q-value $Q(s, a)$ and the new sample estimate into a running average to update the corresponding Q-value.

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha[sample]$$

α refers to the learning rate. In our case, we initialize α to be 0.5 so that it weighs the new sample and the old estimate evenly.

Our Q-learning agent also utilizes a decaying $\epsilon - greedy$ algorithm to strike a good balance between exploration and exploitation. We initialize ϵ to be 0.5, meaning that initially it is equally likely for the agent to choose a random note and to choose the note that yields the highest Q-value. Then, we set the decaying factor to be 0.96 so that for every iteration, ϵ gets lower, meaning that it becomes increasingly less likely for the agent to randomly choose an action.

$$\epsilon \leftarrow 0.96 * \epsilon$$

In this way, we not only explore the space, but also ensure that the agent will not thrash around when the learning is done.

C. Q-learning States

Our Q-learning States represents the framework composed by all notes and timestamps information from which the Q-learning agent would be able to select. The information is stored as a python dictionary with structure shown as follows:

- key: (start time, end time)
- item: (root note pitch, [all new note pitches])

Information represented by the dictionary is initialized as the files encoding music generated by Magenta are read into the Q-learning python file. The dictionary utilizes the timestamp information, a tuple of floats representing the start time and end time, as keys, and saves a tuple that consists the pitch of the root note and a list of pitches of the new notes (generated by PolyRNN) as items. All new notes will correspondingly start later than and end earlier than the start time and end time information designated by its key. Each Q-learning State utilizes one timestamp. The State is initialized with exclusively the start time and end time of the timestamp, the pitch of the root note, and all pitches of the new notes in such timestamp. The Q-learning agent would travel through each State with the ability to access the information of the previous and next States as well as choosing the best rewarded new notes from all new notes provided.

D. Reward Function

We utilized music theory to design a reward function. The reward function is defined to be the maximum of a reward measuring how much the melody is in major key and a reward measuring how much the melody is in minor key.

$$\text{reward} = max(\text{major\_reward, minor\_reward})$$

We define a root note to be any note from the original monophonic melody, and we define a chosen note to be any note our algorithm has picked to form a chord with the corresponding root note. Each reward consists of two components, where the first one is called a melodic reward, measuring how melodic each accompanying note is when it forms a chord with the root note at each timestamp. The second part is called a comparison reward, measuring how the chosen note adds dynamics to the song given the previously chosen notes.

$$\text{major\_reward} = \text{melodic\_reward} + \text{comparison\_reward}$$
$$\text{minor\_reward} = \text{melodic\_reward} + \text{comparison\_reward}$$

Our main goal is to reward generated consonant notes and penalize dissonant notes. At each State, we generate a major chord and a minor chord based on the corresponding root note at that State. Greater reward will be given if the chosen note is part of the chord, and the reward will be added to either major_reward or minor_reward based on which chord is used.

On the other hand, we harshly penalize notes that are one half step or one whole step away from the root note because they form a dissonant melody. If the chosen note is the same as the original note or is identical to the chosen note at the previous timestamp, then a moderate penalty will be given. In addition, notes one octave away from the root notes will also be given a negative reward because large separation destroys the melody of chords. Finally, other combination of chords that are acceptable yet not dissonant will be given a small positive rewards.

E. Training and Testing

We trained our model using 4295 monophonic melodies. In order to make our model memorize past experience and apply it to later one, we utilized the concept of transfer learning: after each music piece is trained for 100 iterations, the final result is recorded in a probabilistic four-dimensional dictionary structure that provides information about the probability that any notes will be chosen when given a certain root note, its previous root note, and its previous chosen note.

After training, we applied our model to the testing dataset of 477 files and evaluated the results both objectively and subjectively by using an evaluation metric and by asking volunteers to describe the enjoyability respectively.

III. Results

A. Development of Evaluation Metrics

Different from other machine learning tasks such as classification and prediction, music evaluation is an inherently subjective process. Nevertheless, to evaluate the performance of our AI, a relatively objective measure is necessary. Therefore, we researched evaluation metrics that other researchers have utilized in their studies. We found that in addition to pitch and timing, metrics measuring coherency, novelty, and harmony appear frequently in these studies [1] [8]. Considering that in our case, pitch is the only variable, we selected coherency, novelty, and harmony to be the major evaluation criteria. We established a scoring function to reflect these characteristics of each music piece.

We initialized the score of each piece to be 100.0 and deduct points whenever violations of good music practices are found.

1) Coherency: Harmonic music usually exhibits coherency in each line of melody. When two adjacent notes are too far away from each other, the large jump could result in incoherence in the melody, consequently destroying the satisfactory flow of the piece. Therefore, in order to maintain the notion of coherency, we calculated the average of absolute distances between all the adjacent notes and subtracted the value from the overall score. In this way, we ensured that no two notes are too far apart.

2) Novelty: Although repetition is a common characteristic in music creation, our melody only consists of eight notes, making repetition a less optimistic characteristic in our case. Therefore, in order to maintain novelty, we penalized chosen notes that are the same as the root note. Meanwhile, we also penalized chosen notes that are too far away from the root note. These notes do not add any new artistic value to the music piece. Whenever such notes are found, we subtracted 5 points from the score.

3) Harmony: Harmony is arguably one of the most important features of good music. Even one dissonant note could destroy the whole piece. Hence, we first defined the dissonant notes based on each root note and subtracted 10 points from the overall score whenever a chosen note is in the set of dissonant notes. The magnitude of penalty is higher than other criterion considering the damage a dissonant note could do to our overall piece of music.

B. Distribution of the Final Scores

We split the music files into the training set and the testing set. Out of the 4772 samples, 90% of the music files were used as the training samples, while the rest 10% were used as the testing samples. We first ran PolyRNN and Q-learning on the training samples and simultaneously updated our probabilistic table for transfer learning. Then, we applied the generated probabilistic table to the testing samples and gave them corresponding scores based on the metrics we defined above. Out of 477 testing files, we have the following distribution.

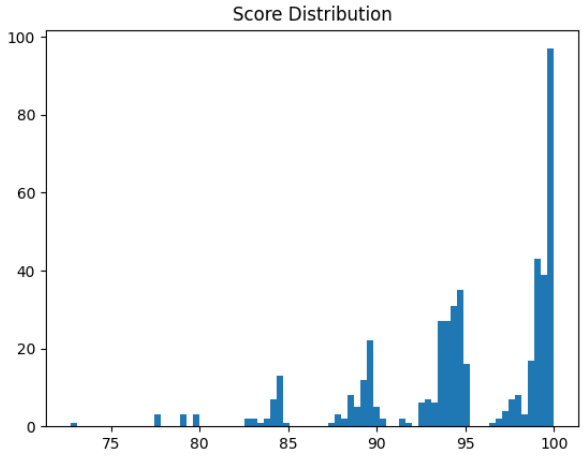

Fig. 2. Score distribution of the testing samples.

The mean of the scores is 94.97, while the standard deviation of the scores is 5.13. It means that the music pieces generated indeed have relatively high scores. However, as shown in Fig.1, there are several clusters in the score distribution. We looked further into these gaps and found that harmony is the factor that leads to these gaps. For each cluster, the lower the score is, the more dissonant notes occur in the piece. In each cluster, the coherency factor mainly contributes to the deviation. More coherent pieces have relatively higher scores.

C. A Subjective Point of View

As discussed before, music evaluation is inherently subjective. To ensure that our scoring function makes sense in

the real world, we picked generated music pieces from each cluster, showed them to several friends, and asked them to rank the pieces based on how good it sounds. The pieces we chose have scores $100.0, 95.0, 90.0, 84.8, 72.7$ respectively. Surprisingly, all the people ranked the piece with score of 100.0 as the best piece and the piece with score of 72.7 as the worst piece. It appears that our scoring metric successfully captured the essence of the "best" music and the "worst" music. The pieces with other scores, on the other hand, received mixed reviews. Five people ranked the piece with score of 84.8 as the second best piece, while the rest seven ranked it as the second worst piece. The subjectivity of music comes in here. It is hard to tell whether the discrepancies result from the different taste of music or the defects in our scoring metrics.

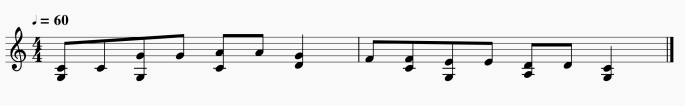

Fig. 3. Generated Polyphonic Melody with score of 100.0

IV. Conclusion

Our result suggests that harmonization of melody can be achieved by artificial intelligence by using a combination of RNN models, Q-learning, and transfer learning. As shown by the average of the evaluation score 94.97 and the feedback from our volunteers, we have successfully created a model to produce melodic polyphonic music. Our automatic music AI can be utilized to help music learners with music composition, especially with duet composition. However, music produced by our AI is not yet comparable to human generated music, as it cannot create as many dynamics, moods, and hidden messages like famous musicians do. Consequently, even though this model can serve as a shortcut to music production, we recommend this model to be used as a reference or inspiration.

V. Limitations

The project shows four aspects of limitations that can be potentially improved in future work: the reward function, the evaluation metrics, the scalability and the generalizability, and the dataset.

A. Reward Function

The current version of reward function assigns reward or punishment according to specific notes compositions at some specific timestamp. Even though it considers the continuity of notes for two adjacent timestamps, it is simply following a precise however partial manner, for there could be circumstances that cannot be captured by such mechanisms. People accept music for their diverse and flowery composition of notes and styles, and the standard for a beautify piece of music extends far beyond the major chords and minor chords. For example, a human-made polyphony will possibly fill the gap in the main melody by inserting appropriate notes into the sub-melody, while such operation is impossible for this AI-generated polyphony since there are currently no viable reward function that supports selecting notes to put between a gap. Therefore, a better reward function that covers as many circumstances as possible using a smart strategy (preferably not enumeration) would be even better for this project.

B. Evaluation Metrics

The evaluation metrics also need further examination, for it is unclear whether the evaluation metrics actually produce an objective score that represents the beauty of the music. This will be difficult, for it is highly debatable whether the evaluation of music itself can be quantifiable due to the fact that different individuals can possess various opinions regarding a single line of music. Also, the current evaluation metrics share a similar problem with the reward function as it is also covering a limited number of circumstances. More situations should be included and considered to produce a relatively authoritative score.

C. Scalability and Generalizability

The scalability of the project reflects a problem with the short length of the music pieces in the training sample, which all have only 8 seconds long and usually contain fewer than 16 notes. It is doubted whether the model still performs as usual when the input melody is longer (for example, the whole main melody of a symphony), for it might require smarter algorithm to accelerate its running speed. The generalizability refers to the ability of the model to adapt to different styles of music. The current training samples are all from the Bach-Doodle dataset, therefore sharing the music style of Johann Sebastian Bach. An adaptive model would preferably train its parameters so that its reward functions and final evaluation metrics behave according to the rules learnt from the training sample.

D. Dataset

Bach-Doodle dataset contains MIDI file of the user-entered melody. However, since users only have up to 16 notes to fill in, the lack of room for creation inherently results in repetition of the music pieces. In other words, many music monophonic pieces we used as inputs have similar melodies, which can unintentionally make our Artificial Intelligence agent learn a certain pattern and therefore result in a biased model.

VI. Future Work

The chief future work would be to improve the transfer learning process of the model. The current model simply stores a matrix containing the overall probability of each possible combination of notes from current and previous timestamps, therefore it is inevitable that the trained model arrives at the same final probability for several circumstances. Such cases illustrate the necessity of developing a method of choosing the best notes from all candidate notes sharing the same probability. Also, it would be necessary to investigate other transfer learning methods and apply those methods into the current model. The current transfer learning method has

high space complexity as well as time complexity as it is parsing through a four-dimensional matrix with around 80 items in each dimension, therefore would seriously demand high computational power. A transfer learning method with better algorithm and lower complexity would be better for this project.

ACKNOWLEDGMENT

We would like to take this opportunity to explain the distribution of our work:

- Mint Lin: Design and implementation of the reward function
- Yuchen Sun: Data processing and transformation
- Simon Zhu: Implementation of Q-learning agent

We collectively did research on dataset and models, implemented Q-learning and transfer learning, and authored the report.

We express gratitude for Professor Lu Feng at the University of Virginia for assisting us in this project.

REFERENCES

[1] C.-Z. A. Huang et al., "Music Transformer," arXiv:1809.04281 [cs, eess, stat], Dec. 2018, Accessed: Nov. 30, 2020. [Online]. Available: http://arxiv.org/abs/1809.04281.

[2] "Research," Magenta. https://magenta.tensorflow.org/research/ (accessed Nov. 30, 2020).

[3] "magenta/magenta," GitHub. https://github.com/magenta/magenta (accessed Dec. 02, 2020).

[4] Juan P. Braga Brum, "Traditional Flute Dataset for Score Alignment," 2018. https://kaggle.com/jbraga/traditional-flute-dataset (accessed Dec. 02, 2020).

[5] J. Gillick, A. Roberts, J. Engel, D. Eck, and D. Bamman, "Learning to Groove with Inverse Sequence Transformations," arXiv:1905.06118 [cs, eess, stat], Jul. 2019, Accessed: Dec. 02, 2020. [Online]. Available: http://arxiv.org/abs/1905.06118.

[6] C. Hawthorne et al., "Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset," arXiv:1810.12247 [cs, eess, stat], Jan. 2019, Accessed: Dec. 02, 2020. [Online]. Available: http://arxiv.org/abs/1810.12247.

[7] "Bach Doodle Dataset," Magenta. https://magenta.tensorflow.org/datasets/bach-doodle (accessed Dec. 02, 2020).

[8] N. Jaques, S. Gu, R. E. Turner, and D. Eck, "Tuning Recurrent Neural Networks with Reinforcement Learning," Nov. 2016, Accessed: Nov. 30, 2020. [Online]. Available: https://arxiv.org/abs/1611.02796v3.