Fine-tuning a polyphonic music generation algorithm using reinforcement learning

Vasily Inkovskiy
Carleton University
vasilyinkovskiy@cmail.carleton.ca
Aidan MacGillivary
Carleton University
aidanmacgillivary@cmail.carleton.ca
Angus MacGillivary
Carleton University
angusmacgillivary@cmail.carleton.ca
Fraser Rankin
Carleton University
fraserrankin@cmail.carleton.ca
Zhi Ou Yang
Carleton University
zhiouyang@cmail.carleton.ca

December 8, 2024

0.1 Approaches used in this project

0.1.1 Simplified Monophonic Q-Learning Approach

The first approach used in this project is a Q-learning and temporal difference (TD) approach. It's typical to use deep Q-learning, a neural network approach, for this type of application due to the high number of states. However, we used a standard Q-learning approach as deep Q-learning was not covered in the course material and this serves as a starting point for this project. While not ideal, it is still applicable because the environment can be simplified down to a feasible scale. As a result of this approach, it is a very simplified way of creating music compared to how a professional artist would typically create music. It is monophonic (one sound at a time), has only a range of one octave, has a pre-defined key set to the lowest note, and its representation of state is heavily simplified. All these attributes contribute to lowering the number of possible states to a point where it is feasible with Q-learning. Due to the complexity of music, it can otherwise quickly turn into an unmanageable amount of states. This approach simply defines the state as what note was last played, when it was played, the current time in the episode, and whether a note is currently being held or not. While this definition of state does not capture the entire composition, it is still capable of a lot; it can reward certain pitches, timing, and length of notes. The major sacrifice that comes with not having access to the entire composition is not being able to identify and replay motifs, which is a cornerstone in most pieces of music.

While training, the agent uses an epsilon-greedy policy to build its Q-table. This policy makes it focus most of its information gathering on relevant actions that are being rewarded, but also explore other actions on rare occasion to avoid tunnelvision. This is off-policy training though. The target policy, which is used for the final episode, is a softmax-like policy.

It calculates a distribution of probabilities for each action weighted by its Q-value then randomly selects one. Additionally before it does this, it filters all available actions into a “good” list and randomly chooses from those. It does this by relatively comparing each Q-value in the current state to an arbitrarily-decided threshold. The reason for making the target policy function this way is because exploiting or exploring in a traditional sense have major down sides for this application. Exploiting causes over-exploitation where the agent simply selects the same notes repeatedly and fails to generate unique or interesting melodies. Exploring and choosing a random action causes the agent to occasionally play a note that is out-of-key even though it has been significantly penalized for those actions during its training. So, this target policy aims to have the agent randomly select an action, but only from a set of actions that have been proven to perform well enough during previous episodes. This achieves randomness while still playing mostly in-key.

Without a reward structure, this method simply selects the first action each step. With how the actions are designated, this would mean repeatedly playing sixteenth notes at the lowest pitch each step except for the occasional random action due to exploration. So, the environment needs to lead the agent toward more acceptable compositions that would sound pleasant. This is achieved through various music theory rules:

- **Play notes that are in-key**

  This is done by assigning arbitrary reward values to each possible pitch that is able to be played. Off-key pitches are heavily penalized while in-key pitches are barely penalized. There are also slightly higher rewards assigned to more consonant or “good-sounding” notes such as the tonic and dominant.

- **Maintain appropriate ratios between play, sustain, and rest actions**

  This is achieved by simply increasing and decreasing the reward for each action appropriately. For example, the rest action has a default penalty of -3 while a play action has a default penalty of -4, incentivizing the agent to rest slightly more than it plays notes.

- **Avoid playing the same pitch repeatedly**

  This is done by comparing the current action with the pitch of the last note, which is stored in the state. If the pitches are equal, the agent is penalized.

- **Keep the average note length to about one beat**

  This is made possible by the fact that the state keeps track of the last note, the current step its on for the episode, and whether a note is currently being played or not. The duration of the current note can be calculated by subtracting the step the last note was played from the current step then rewarding or penalizing the sustain action accordingly. If the last note was played less than four steps ago, reward the sustain action, otherwise penalize it.

- **Play at a somewhat consistent rhythm**

  This rule is already supported by the one above, but it is enhanced even more by rewarding notes that are played on steps that are a multiple of four, meaning that it was played at the start of the bar, assuming a 4/4 time signature. We also slightly reward notes that are played right before an on-beat, known as pickup notes. Pickup notes help add variety in the timing and length of notes.

- **End each episode with resolution**

  This is easily done by checking the last note's pitch at the end of the episode. If it is the tonic, then a significant reward is given. The tonic is the first note of a key and is known for resolving tension in music; it "feels like home".