EECS 395 Project Proposal for Rhythm Game Charter AI

[redacted]

Technical Project Requirements and Specifications

Create a program that can generate rhythm game files when given an audio file and a difficulty setting. Identifying notes from raw audio samples is a difficult task because location, accuracy, and type of note must be calculated, and the sequence of notes that came before it must be taken into consideration. We will attempt to use two neural networks to 1) calculate step location and 2) choose the step type for each location.

Proposed Solutions

In general, the project will contain 4 components wrapped in a single easy to call API. Each of these sections is detailed below:

1. Audio Preprocessor
2. NN1 Find step placement*
3. NN2 Find correct step for each valid location*
4. Convert output of NN1 and NN2 into chart format that can be used in game

* A “step” in a rhythm game is an entity that the player must interact with in order to “play” the music as the game progresses through the song. In DDR, steps are arrows that indicate where the player should put their foot down next. In Guitar hero, steps are the notes that travel towards the player on a fretboard and that line up the buttons on the guitar controller.

Audio Preprocessor

We infer that it would be beneficial for both NN1 and NN2 that rather than passing in the raw audio to both, we would first run the audio file through an algorithm with the goal of:

1. Normalize - For most consistent results, data passed into our NNs should have similar max amplitudes.
2. Remove noise - this kind of functionality is already available in open source libraries https://pypi.org/project/noisereduce/
    a. Optional component which will be added / removed depending on its effect on the quality of the output of NN1 and NN2.
3. Generate BPM info - if possible, we should calculate the BPM of the audio files as well as any changes that occur as the audio progresses. This data can also be passed into the start of the algorithm if the end user already knows the BPM data

    a. BPM data can be a set constant or a list of BPM changes with timestamps
    b. There are machine learning methods (neural networks) designed to determine BPM that may be more accurate than traditional methods.
4. Generate Intermediate File Type - Should the normalized, denoised data not generalize well when fed through our neural network pipeline, an intermediate data representation, such as a spectrogram, can provide better means for analyzing the audio data.

Neural Network 1

Identify location of each note, considering which notes to keep (1/32, 1/64) within a measure, based on audio information after preprocessing. Depending on level, there should be more or less steps placed in a measure.

- Possibly use spectral flux to determine onsets (when a note starts)
- Each difficulty level has statistical restrictions on the ability to place notes on the beats that are not subsets of the general time signature.
    - For example: in a 4/4 time signature, the "easy" notes are placed at the 1/4th 1/8th and possibly 1/16th notes. As the difficulty gets harder, the 1/32nd and even the 1/12th notes become available for step placement.
- Given BPM info, the NN will be able to look for valid notes at specific intervals since it is known that a note should be available there

Neural Network 2

Identify what kind of step to place at each location found by NN1 using a GAN NN. This is determined by several qualitative factors, but also is heavily influenced by the whims of the NN. Often there isn't an exact or best solution for step placement, however humans are often able to discern if the step placement feels "natural" which is our goal regarding the output of this NN. Especially in games like guitar hero, pitch progression plays an important role in step placement. Though even then, tens of tones must still be represented as only 5 buttons.

We might not necessarily use a GAN for all games. If we find that a GAN is not producing good results, we could potentially replace it with a simpler training system, comparing with the training data and augmented by rules. For example, some user defined rules could be:

- OSU
    - The max / minimum radius or arcs
    - Min distance between chained notes
- DDR / Guitar Hero
    - Min duration of held notes (easier difficulty = longer held notes)
- All
    - Max occurrences of simultaneous notes

    - Minimum duration between notes can be enforced (this is something that should be learned by the GAN, but if defined programmatically will kick start the learning process)

Each game has a different concept of what a step is, but there are a few consistencies across most rhythm games step types:

1. Instant beat - the step happens at a set time and has 0 duration.
2. Hold and release - the step happens at a set time and must be interacted with for some duration.
3. Position - this concept is different for all rhythm games, specific examples are:
    a. DDR - One of the four directions step attributed to
    b. OSU / Beatsaber - Where on the screen (x y coordinates) notes appear, (OSU only - and for hold notes, curves/directions they move)
    c. Guitar hero - On which note of the fretboard is the step occurring
4. Simultaneous steps - This is heavily influenced by the difficulty parameter provided at the beginning of the algorithm. Easy modes will never have more than a single note pressed at once, with the exception of very few Guitar hero songs and some DDR charts. As the difficulty level increases, multiple simultaneous notes can often be encouraged even for a monotone note

Convert Output

We will be outputting data in a relatively simple format, so that output must then be converted to a format consumable by the game specified along with the audio file at the start of the algorithm. Every game has a different format for the files processed. Several games such as Stepmania and Osu already have extensive official documentation on their file format online. Other games such as Beat Saber and Clone Hero have unofficial documentation written by users. As such, we do not expect significant difficulty in converting the direct output of the neural networks to the individual file formats. This could be avoided by having the neural networks operate on a per-character output basis, but that seems likely to be significantly harder to train for little benefit.

Background

- Familiarity with neural networks frameworks
    - Tensorflow / Keras / PyTorch
- Rhythm games and their chart formats
    - Currently planning support for
        - Stepmania (.sm files, an open source DDR clone)
            - https://github.com/stepmania/stepmania/wiki/sm

        - OSU (.osu files)
            - https://osu.ppy.sh/help/wiki/osu!_File_Formats/Osu_(file_format)
        - Beat Saber (info.dat and <difficulty>.dat files)
            - https://github.com/Kylemc1413/SongCore
        - Clone Hero (Guitar Hero clone)
            - This one we may not get to
            - https://clone-hero.fandom.com/wiki/Charting
- Python programming language
    - Libraries - numpy, pandas (including NN frameworks)

Hardware Requirements

Requirements marked with a star (*) are to improve game experience, but a keyboard can be used as a substitute.

- DDR Mat (Stepmania)*
- Guitar Hero Guitar (Clone Hero)*
- Virtual Reality System (Beatsaber)
- Nvidia Graphics Card / HPC Access (Neural Network training)

Related work

- Dance Dance Convolution (2017)
    - An academic paper with similar goals and methodology, but focused only on DDR and with a corpus of only ~200 songs
    - https://arxiv.org/abs/1703.06891
    - https://github.com/chrisdonahue/ddc
- Musical Beat Detection with Convolutional Neural Networks (2016)
    - A comprehensive blog post on a similar goal, but on an even simpler game
    - http://tommymullaney.com/projects/rhythm-games-neural-networks

Work to be Done

In the preliminary stages of this project we will need to decide exactly what information goes into the generic file produced. Since different rhythm games have different information requirements, we really want only the information that would make sense for a majority of them. Once we have that, we will need to collect as many example charts as we can (where “chart” refers to whichever file format each individual game uses) and convert them into this generic format. We will then have to create and train the neural networks to process the music into the generic format, and then the generic format into each game’s end result. The games’ files will also need to be processed into some sort of format which is easy for the neural networks to output, as we will want to avoid making the networks learn to output proper header data.

Management Plan

To ensure the completion of this project, we will set rough weekly goals to be completed by the team. These goals are not concrete, given the tumultuous nature of training deep neural networks, but these goals should be met in a timely manner.

Deadlines:

- ❏ Week 2 - Discuss ideas, explore related works, formulate plan.
- ❏ Week 3 - Submit Project Proposal Document
- ❏ Week 4 - Meet with Professor Xu, refine proposal.
- ❏ Week 5 - Acquire and preprocess data.
- ❏ Week 6 - Generate audio file pipeline.
- ❏ Week 7 - Train/Debug models (DDR only).
- ❏ Week 8 - Train/Debug models (DDR only).
- ❏ Week 9 - Train/Debug models (all games).
- ❏ Week 10 - Continue debugging if needed, acquire more data if needed, playtest all DDR mappings.
- ❏ Week 11 - Playtest outputs for all games considered.
- ❏ Week 12 - Playtest project output with friends.
- ❏ Week 13 - Implement final changes.
- ❏ Week 14 - Prepare poster for intersections.
- ❏ Week 15 - Submit final project.
- ❏ Week 16 - Demonstrate final presentation and demo project.