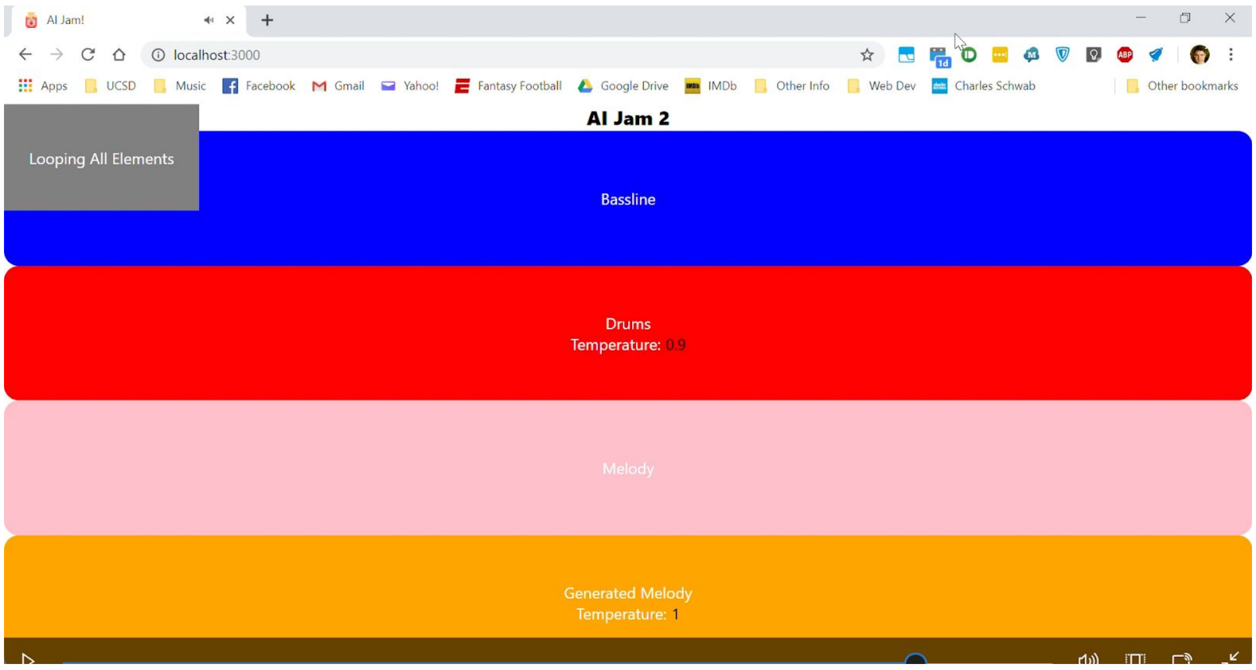
Machine Learning for the Arts
UCSD SPRING 2019
FINAL PROJECT

AI Jam 2



Matthew Rice

DESCRIPTION

Concept:
I was inspired to create this project by my experience with Google's Magenta projects, specifically the AI Jam (demo at https://www.youtube.com/watch?v=QlVoR1jQrPk, repo at https://github.com/tensorflow/magenta-demos/tree/master/ai-jam-js) project as well as the new Magenta Studio for Ableton project. I found the AI Jam project interesting, but I wanted to expand on the idea. Instead of having the AI improvise one melody based on your input, I wanted to have the AI generate different parts that I could layer together and improvise with, similar to how musicians at a jam session improvise based on the other layers. In the original AI Jam, there is only a maximum of two parts playing at the same time, but in my version, I had up to five parts playing at the same time.

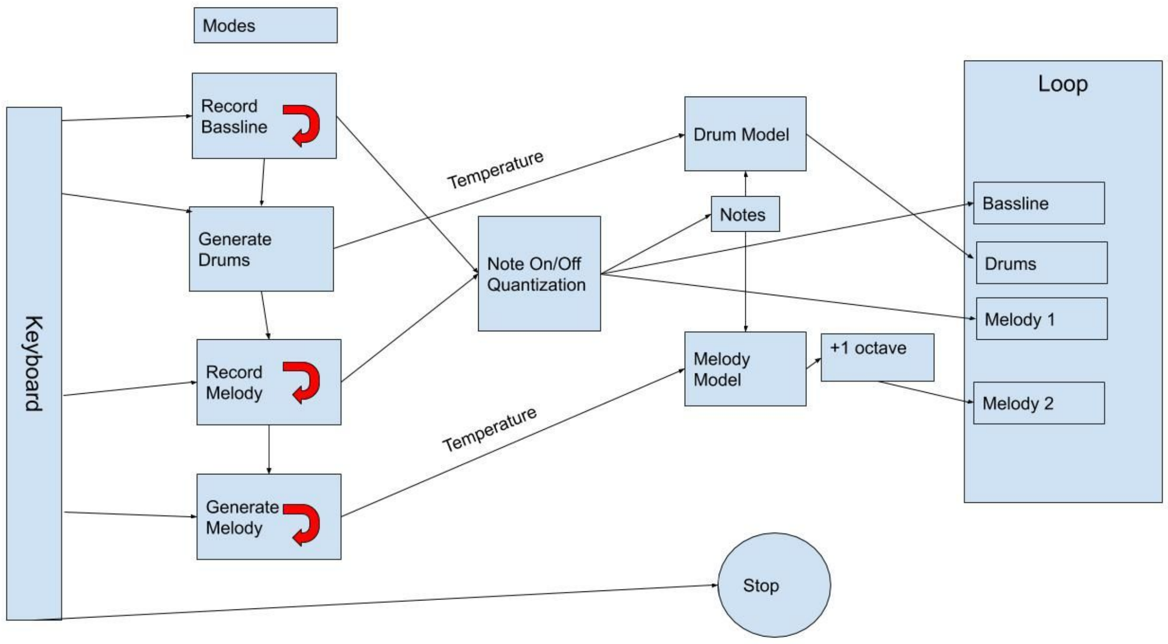
Magenta studio was released in February as a meaningful way for producers, composers, and musicians to add machine learning to the music they are creating. The studio can fully integrate with Ableton Live 10.1, using MaxForLive which is a popular digital audio workstation. Although I didn't have the latest version of Ableton to try Magenta in Ableton, I was able to try a Windows standalone version. The two plugins from this studio that I found the most inspiring were the Drumify plugin and the Continue plugin. In the Drumify plugin, one inputs a stream of notes, which are converted into "taps" by removing the note information, and then a midi drum beat is created. The Continue plugin takes a bass or melody and extends it by generating a bassline or melody that could potentially follow the original.

One of the problems with a lot of music generation techniques seen online is that much of the music generated is not "good". Although one would think that music generation itself is a subjective art, there are many objective parts to music composition that add clarity and coherence, leading to a listener to identify a pattern in it - similar to how most pop music melodies are "hummable". However, from my experience, both of the models used in these plugins work fairly well, especially with quantized inputs, and generate music that is not only interesting, but also "good" and has enough clarity to use in an improvisation session.

These projects showed me ideas about novel ways to integrate the rapidly improving music generation techniques into a usable feature for musicians. However, many of their capabilities weren't real-time. They were designed for you to create something, then as it was looping in Ableton, click a few buttons, have it generate a midi track, and you then had to drag the track onto a new midi track yourself. I wanted to create a demo experience where everything would be in real-time as much as possible, and the musician could solely focus on improvising with the machine, just as it is in the original AI Jam. This led me to create AI Jam 2.

You can write about what inspired you, what the meaning of your work is, what kind of idea or imagination brought you to this project, and/or any other thoughts that are relevant to the idea and/or meaning of your work. If it needs to be supported by any literature review, meaning study of other works in the field or in the history you can add it here too.

Technique:
AI Jam 2 is a web-based project that uses a midi keyboard for controlling the entire process of music creation. While creating the project, I realized that most of the technical aspects of creating this process have nothing to do with machine learning. In fact, the difficulty in this project was taking the data from a midi keyboard and transforming it into data that could be fed into a model while also looping other music elements all in real time. However, due to my background in web and audio programming I was able to overcome all of the challenges that I faced related to this problem. A block diagram of the application is shown below to help the reader understand the dataflow. The basic flow is this: record a 2 measure bassline, then generate some drums, record a 2 measure melody, then generate a new melody, then at any point regenerate new drums or a melody. All these actions are controlled from the keyboard.



This project is written in JavaScript with React.js with help from WebMidi.js, Tone.js, and Magenta.js libraries. After creating an initial web project, I researched how AI Jam, Drumify and Continue were developed to learn about feeding data to their models. Since they are all open source, I was able to easily solve this issue. I then had the challenge of collecting the data from my keyboard and quantizing it in a manner suitable for the magenta models. One of the biggest challenges in any real-time audio application is having an extremely accurate timing system; failing to do so results in both laggy audio as well as un-quantizable audio. For example, I couldn't rely on the computer to give me accurate timings to quantize my audio to, since the browser implementations of timing loops (like setTimeout/setInterval) can have variances greater than 100ms which is extremely slow for audio. Luckily Tone.js in coordination with the Web Audio API has a sample-accurate timing system that allows for near-perfect accuracy in timing. I was able to use Tone.js's Transport system in combination with the midi data I received from WebMidi.js to quantize all incoming notes to a grid of 16th notes at 120 bpm. I then was able to use some time conversions from Tone.js to create JavaScript objects needed to feed the magenta models. I was able to repurpose code from the Drumify and Continue plugins (which used Magenta.js) to feed my quantized data to the models and give me the output data when the models were complete.

In both models, I fed in two measures of data and received two measures of data back. Each model was given a temperature that I initialized to 1, but would decrease by 0.1 on successive generations. The drum model is a tap VAE trained on thousands of drum midis, while the melody model is a LSTM model trained on thousands of monophonic melodies. For the melody model, it was intended to extend a melody, but I would play both melodies at the same time, sometimes creating pleasing harmonies.

Creating the sounds and looping system wasn't too challenging. For the bassline and melody synths, I modified some Tone.js synths, added a Tone.Reverb, and a Tone.Vibrato to some synths. For the drums, I used some drum samples from a drum sample pack I had from my music production software and mapped all the incoming data from the drum model to play the correct drum. Also, while recording, I played a metronome sound (https://freesound.org/people/HollowRiku/sounds/146781/) on each quarter note to help the user record to the grid.

The last part of this technology was creating a simple visual interface for the user to understand which section they were recording, as well as creating a visual front-end to the application. I designed an interface that would show the current state in the jamming process, as well as give the temperature for the drum and melody models. It also flashed an information box while the metronome was clicking.

Process:
While "jamming" with the AI, I discovered a few notes. Firstly, the drum generation seems to work significantly better with a dense bassline instead of a sparse one. I imagine this has to do with the midis it was trained on, as using a sparse bassline frequently generates less than six drum hits in two measures. Secondly, I noticed that the melody generation tends not to change too much using different temperatures, and tends to gravitate towards similar first notes and rhythms. Finally, since all of my generations are monophonic, it can be frustrating to create basslines and melodies using traditional piano techniques, since if two notes are played at the same time, I ignore the latest one in quantizing for the model. This quantization process can make generating the bassline or melody that is in the user's head irritating, since many times notes are ignored. Also, I quantize to 16th notes, which makes swung rhythms or triples not

doable. However, in general the quantize system aids the user and model in developing coherent ideas.

Result:
The final product was the live demo at the presentation day which ultimately was successful. I presented a live interaction with the idea hooked up to a K10 speaker for more pleasing audio. I also created an example mp3 file of a jam session I had done earlier. Also, one can check out a live version on the web at http://mhrice.github.io/Ai-Jam-2.

Reflection:
I'm pretty satisfied with the final result. The jam session was fun and interactive for both myself and others that tried it out. In the future, I plan to create a better interface for playing instead of using custom mappings on my keyboard. I also want to extend this two measure jam into a longer jam session. However, from what I've experienced, the models tend not to be as successful in generating longer musical structures. Perhaps new models can be used for this. Overall my goal of creating a live AI jam session was accomplished and I look forward to using machine learning in new tools that I create for producers and musicians.

CODE: https://github.com/ucsd-ml-arts/ml-art-final2-ai-jam-2