Omnisong, an infinite song generator
MUS 1051 - Fundamentals of Music
2025 Fall Semester

Introduction

During the semester, we have discussed what music is, what it means, and its role in our lives. A common sentiment is that music is, simply, sounds organized in space. But that doesn't answer who can make music. Do we, and should we, care about who is creating music? Is it simply a human expression, or can machines do it too?

One may answer that music is solely human because it requires intentionality and a desire to express something. This would take a lot of computer-generated music out, but there's one technique that cannot be so easily discarded: artificial intelligence. At its core, it is simply a recreation of what goes on in our own brains: you have neurons that connect in neural networks, and together they process new information by adapting themselves and their neighbors to the new stimulus. With the information they have gathered, they can start spewing their own creations. Sounds familiar, right?

This adds a lot of nuance to discussions on musical authorship: with an artificial brain generating music, several patterns can be mixed and matched, creating unique compositions based on whatever the model was trained with. For a lot of human musicians, for example, that would be classical music and other Western concepts, such as chord progressions, key signatures, note patterns, and techniques, and much more. If you add to your training data more meaningful pieces (e.g., a composition that represents a sad moment through chord progressions focused on the minor scale), you could have a model that will generate actual program music.

My project seeks to start challenging these definitions of music by showcasing how a computer can produce music the "same" (at least in the biological sense) way a human would. By training a model on chord progressions and other common music techniques, you can have compositions that make sense within a formal musical context. And, of course, it can do infinitely.

Training Data

```
44166   bm_376 H1_84 H4_78 H7_94
44167   bm_213 H12_108 H11_67 H10_135 H9_82 H8_101 H7_80 H6_138
44168   bm_217 H1_93 H2_84 H1_84 H12_55 H1_112
44169   cm_312 H1_145 H6_96 H11_135 H4_109 H9_72 H2_127 H7_68 H12_121 H5_104
44170   g7_398 H1_120 H2_61 H3_128 H4_109 H12_86 H11_105 H10_78 H9_76 H8_78
44171   cm_397 H1_82 H4_78 H7_55 H10_141 H1_67
44172   am_319 H1_98 H2_120 H3_125 H4_139 H5_125 H6_120 H7_73 H8_132
44173   am_227 H1_129 H2_58 H3_133 H4_85 H5_150 H6_110 H7_103 H8_50 H9_65 H10_110
44174   am_272 H7_126 H3_77 H9_130 H6_127 H4_70 H11_88 H3_108 H7_131 H7_88 H2_51 H8_124
44175   am_208 H12_68 H11_63 H10_59 H9_51 H8_102 H7_80 H6_134 H5_86 H4_64 H3_133
```

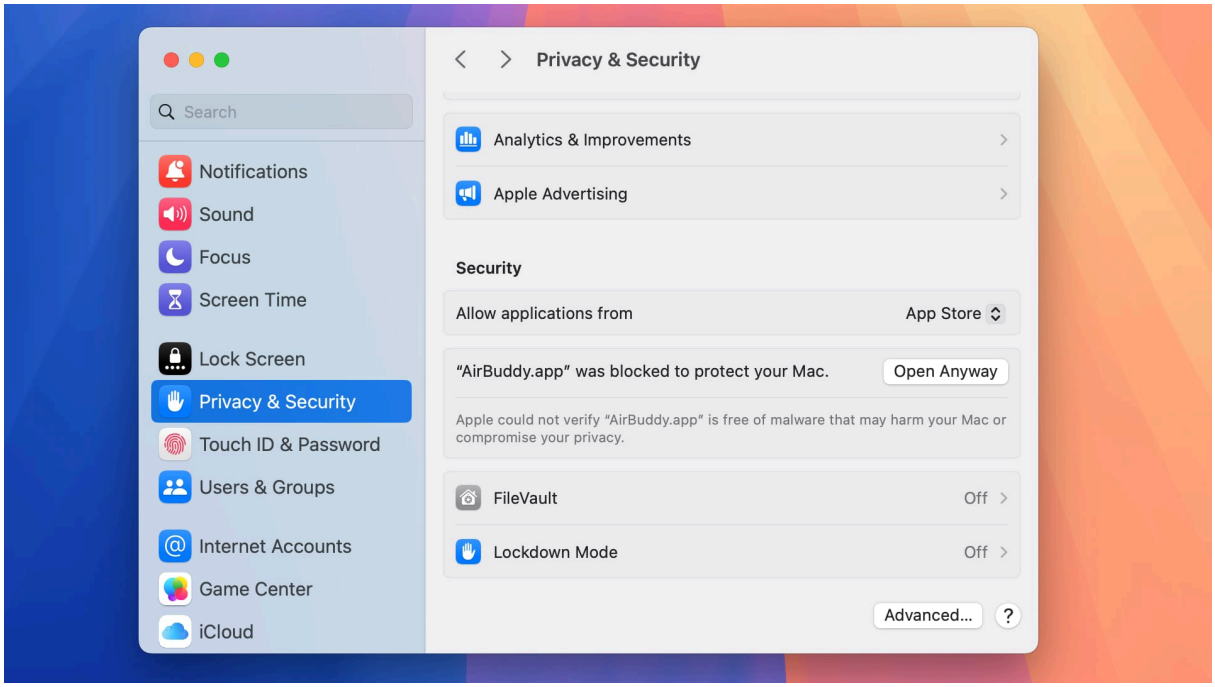
44175 lines of chords and some harp patterns (fifths, wave, descending, ascending, thirds, etc). The chords have been picked based on their function (T/D/PD) and they show up in a valid order.

How to run

Open `Omnisong.app`. Your computer will most likely complain that this file wasn't signed and its safety cannot be assured. If it doesn't say that, yay! Here's an example of said dialogue:

If this screen shows up, head to System Settings > Privacy & Security > scroll to the bottom. You should see something along the lines of "Omnisong.app was blocked to protect your Mac". Click on Open Anyway. I promise it's not a virus - if it is, you know where to find me. This message shows up because the app didn't

go through Apple approval because it costs $100/year, and I have no use for it since I don't develop apps (for now, anyways).



Technologies, collaborators, and specificities

Omnisong is a Python app built with PyQt6, PyTorch, and Pygame, and it uses a GPT (generative pre-trained transformer) model (autoregressive decoder-only transformer). The definition of the model can be found on `model/gpt.py`.

Omnisong was built upon the following technologies:

- The Python programming language. https://python.org.

- PyGame, a Python library for building games. In my case, I used it to play the audio - it has a very nice audio system, which made it easy to load the Omnichord sounds and play them in parallel, just like on the real instrument. https://pygame.org.

- PyTorch, a tensor library optimized for deep learning (in simple English, a library that allows you to create, train, and use models). It's also optimized for Apple platforms (so run the project on an Apple Silicon computer). https://pytorch.org

- PyQt6, a library for building user interfaces using Qt. I chose it instead of PyGame's UI system because it looks like a native macOS app. https://riverbankcomputing.com/software/pyqt/

For people who helped build the project:

- Several random blogs and Medium articles. A non-exhaustive list is:
  https://jaketae.github.io/study/gpt/
  https://medium.com/@akriti.upadhyay/building-custom-gpt-with-pytorch-59e5ba8102d4
  https://medium.com/@minh.hoque/demystifying-neural-network-normalization-techniques-4a21d35b14f8
  https://www.geeksforgeeks.org/machine-learning/activation-functions-neural-networks/
  Etc etc
- ChatGPT was very helpful in making me understand problems with my architecture at the beginning. It also guided me through ways to enhance my model (such as the grammar mask used, the normalization layers I had to add, etc).

Known issues, pitfalls, and considerations

Obviously, this is not a final, perfect work. This project has some limitations because of some of my design choices, which valued development speed and actual feasibility, given the time I had. One example:

The model cannot "hold" a harp note for a random period of time. All of the instructions it was trained on follow a very simple format, such as:

```
g_249 H1_75 H2_143 H1_113 H12_59 H1_82 H2_90
[func=Dominant, type=major, pattern=wave]
```

This just means: play the G chord for 249ms, play the first harp note for 75 seconds, play the second harp note for 143ms, and so on (also, the *func=Dominant* part is not used - it's just a comment added by the script that generates the training data). These are called "tokens" - you can think of them as words. The problem is that the model cannot output words it hasn't seen before. So, for example, if there's no G chord being played for 200ms, there won't be a G chord being played for 200ms. This also means that there's no way to control the time each component of the composition is played.

A way to solve it would be to not use milliseconds at all, but regular note time definitions such as whole, half, quarter, eighth, sixteenth, etc. If you do that, you can control the tempo using the app and set a time signature.

Conclusion

If you would like a more in-depth explanation of how everything works, just let me know! I would be honored to go through my code and what it does (although I may not have a perfect explanation because AI is math and I dislike math). PowerPoint slides would be involved, certainly. Also, this project is open-source & available at https://github.com/lyricalsoul/omnisong. Some components are missing for not being decent enough, such as the training data generator - I can work on that, but I would first find a way to change the vocabulary used by the model so I can have a more dynamic and flexible set of tokens, enhancing generation quality. This document will also be available on the root of the repository.