Is it possible to programmatically generate Vaporwave?

Navan Chauhan

March 14, 2020

1 Introduction

Vaporwave (Stylised with full-width characters as ＶＡＰＯＲＷＡＶＥ) is an ironic variant of chillwave which emerged in the early 2010s as an interdnet meme. [1] It is defined by its slowing down samples of 1980s songs, excessive use of reverb and choppy nature. Unlike other Internet genres, Vaporwave has existed for many years. The heavy emphasis on making itself sound like a production from the 1980s and 1990s differentiates it from other genres. One of it's most characteristic features is the use of samples from "elevator music". Vaporwave is said to be a satirical commentary on the consumerist culture of the 1980s. The discussion of the political ideologies of vaporwave is not the purpose of this article. [2] Programmatically generated music has been on the rise, with some methods using neural networks to analyse and reproduce tunes. OpenAI's MuseNet uses a deep neural network to create 4-minute musical compositions. [3] The approach taken in this article differs from traditional machine learning-oriented approaches as vaporwave heavily relies on remixing rather than creating original content. Therefore the track generated needs to be a remix of an original existing song and thus eliminates the need to create original musical compositions to work with.

2 Blueprint for Vaporwave

”リサフランク420 / 現代のコンピュー” the second track of the Macintosh Plus album released by American electronic musician Vekrtroid under the alias Macintosh Plus largely cemented the blueprint for what a Vaporwave should be. A vaporwave track relies heavily on the sample used, generally, a Rhythm and Blues track from the 1980s is used, after that the song is slowed down until the vocals start sounding eerily. Then an optional pitch correction is done on the track after which beats analysis is done. In the end, special effects, like static Vinyl or Cassette sounds are added to give a sense of synthetic nostalgia. This blueprint can be broken down into the following steps:

- Slow down the sample
- Extract and chop up beats from the song

- Extract and duplicate chorus
- Add reverb
- Add bass and treble

3 Vaporwave Generator

These steps were then turned into a simple script using Python, the code is largely derived from WJLiddy's original repository on GitHub titled "Macintech" [4]. The code was then patched to work on macOS.

3.1 Generator Workflow

The script first fetches the raw audio sample of the song given as input. Then the script slows it down by using Sound eXchange (abbreviated SoX) a cross-platform command-line utility for processing sound, after which it uses aubio (a tool designed for the extraction of annotations from audio signals) for the extraction of beats from the audio file. After extracting the beats, the beats are grouped as lists of 4, 6, 8 or 9 which are perfect for looping. Once this has been done, these groups of beats are randomly duplicated and sections of loops are created. These sections of loops are then randomly rearranged to give the effect of a chopped up track. Finally, a reverb effect is applied on the track and exported as a Vaporwave track.

4 Results and Conclusion

The program was used to generate three remixes of three different songs and these were published on Bandlab: Generated Album

The fact that this was done without actually editing using a digital audio workstation is a testament to the fact that we can generate vaporwave programmatically and we can conclude that it is indeed possible to programmatically generate vaporwave tracks.

The code is available on GitHub navanchauhan/VaporwaveGenerator

References

[1] Born, Georgina; Haworth, Christopher (2017) *Mixing It: Digital Ethnography and Online Research Methods–A Tale of Two Global Digital Music Genres.* pp. 79-81 The Routledge Companion to Digital Ethnography

[2] Whelan, Andrew; Raphaël Nowak. *Vaporwave Is (Not) a Critique Capitalism': Genre Work in An Online Music Scene.* pp. 451-462 Capitalist Aesthetics, special issue of Open Cultural Studies 2 (2018).

[3] Payne, Christine. *MuseNet, 2019.* `https://openai.com/blog/musenet`

[4] WJLiddy `https://github.com/WJLiddy/Macintech`