Singing Voice Synthesis of Latin Chants

Liam Pond — Final Project, MUMT 618

Singing voice synthesis (SVS) is an emerging field that involves the creation of realistic, computer-generated singing. The goal is to produce vocals that mimic human-like qualities, such as pitch, tone, rhythm, and expressiveness, often with the flexibility to sing lyrics in various languages or styles. Using a dataset created from recordings of my own voice singing Latin chants, I trained a machine-learning model to sing previously unseen chants in the style of my own voice. Unfortunately, while numerous obstacles have ultimately prevented the model from synthesizing audio as of December 2024, future work will continue in the coming months to realize the tool's potential. This document will outline the theory, approach, and difficulties encountered.

The impetus for this project comes from Cantus Ultimus [1], a large publicly available database of digitized medieval chant manuscripts. Each of the chants has been transcribed into modern notation, with the words annotated below. However, the existing audio playback feature is low-quality, does not sound like a human, and worst of all, cannot pronounce consonants. The resulting sequences of pitched vowels leave much to be desired, and so this project was born out of an effort to improve the feature. Given that the database is continually expanding, the tool should ideally function automatically, requiring minimal human effort to generate audio.

For this, I chose DiffSinger [2], a state-of-the-art diffusion-based machine learning framework for singing voice synthesis that was developed in 2022 by researchers from Zhejiang University. Like most SVS tools, DiffSinger is language-independent, meaning that it will learn whatever language it is given, and in fact does not even know what language it is synthesizing.

Furthermore, DiffSinger's ease of use and streamlined training process make it an accessible choice for hobbyists wanting to clone their own voice without needing to understand the technical details of machine learning. As a result, DiffSinger has a robust community of passionate users who are continuously innovating and sharing their work on platforms like YouTube and Reddit. While most users are from Japan or China, in the West, the fanbase is also active on the DiffSinger Discord server, with over 1100 members as of December 2024.

At the crossroads of technology and art, the DiffSinger community is a fascinating and unique scene that is fundamentally tied to the internet and is deeply influenced by anime and Asian pop culture. In recent years, Yamaha's Vocaloid has become increasingly popular, with characters like Hatsune Miku becoming iconic digital brands and performing at globally recognizable festivals like Coachella. Similarly, enthusiasts often create unique characters with rich backstories and personalities which they can bring to life by crafting songs and covers with DiffSinger. Although I was an outsider creating vastly different music from everyone else's and had never heard of DiffSinger before the project, I was warmly welcomed and members of the Discord server provided invaluable support, helping to guide me and troubleshoot the many issues I encountered along the way.

Building the Model

The process of building the singing voice synthesis model is straightforward and only requires two inputs: WAV files and LAB files. The WAV files contain the raw audio recordings, while the LAB files provide the timestamps associated with each phoneme. From there, the model can be trained, and new songs can be generated with the same timing, pronunciation, and expressiveness of the original dataset.

The first step is to build the ground truth database. For this, I recorded myself singing six of the chants from Cantus Ultimus, lasting just under 10 minutes in total. According to experienced users, just five to fifteen minutes should suffice with improvements in quality dropping off steadily after that point, although some choose to build databases with several hours of music. The most accessible recording space was my room, which, unfortunately, was not soundproofed and has some echo due to its boxy nature. While I was initially skeptical about this setup, the quality remained high in all the recordings. Through the Audiovisual Hub in Schulich School of Music, I reserved the Zoom H4n recorder, which is portable and did a good job of minimizing unwanted noise. Just to be safe, I used the noise reduction feature in Audacity and put each track through a noise gate to make sure the quiet parts stayed silent.

Once I had my dataset, it was time to annotate. I used a program called vLabeler, which is built specifically for this task, streamlining the process of adding precise labels to WAV files. Though the software was efficient, precisely annotating is an extremely time-consuming process, made worse (and quite frustrating) by lag in the audio playback, which I eventually realized was due to my Bluetooth headphones, although sound from other applications was lag-free. Once each vowel, consonant, breath, and pause for the entire dataset was marked, I exported the LAB files and was ready to begin training.

The training process was straightforward thanks to a community-built Jupyter notebook that provided detailed code samples for each step. However, some of the cells did not run and were missing required imports or had undefined variables, which fortunately I was able to fix on my own. The code was well commented, although many of the comments were not particularly inspiring...

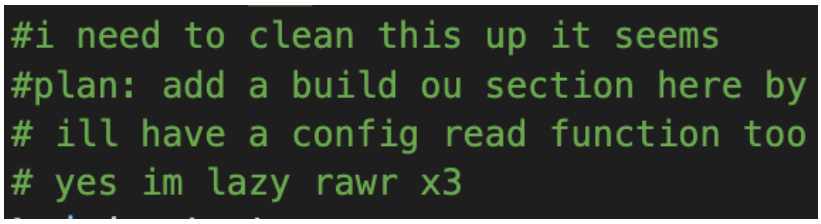

Figure 1. A particularly uninspiring comment

It was at this point that I discovered that the LAB file for my longest song was improperly encoded and could not be read by the computer. The error message indicates that this is usually the case when non-ASCII characters are embedded in the file, although this was not the case. Ultimately, I was unable to determine the cause of this issue, especially given that the other LAB files were fully functional, and so I decided to remove this song from the dataset.

While time-consuming, the training itself was not too difficult and was run through Google Colab, which offers free cloud computing. I naively trained my model until I ran out of credits, at which point I found out that the model needed to be trained twice. The first time tuned the variance parameters, which represents pitch and timing variations, and the second was for the acoustic parameters, which model the timbre and sonic characteristics of the voice. Fortunately, I was able to move my project to a second Google account and resume where I had left off, again training my model until I ran out of credits. Once the model was trained, its output was converted into the Open Neural Network Exchange (ONNX) format, which encodes machine learning models, and then into an OpenUTAU voicebank, which allows me to create new songs using OpenUTAU.

OpenUTAU is a powerful tool with a piano interface where you can input musical notes, type in the corresponding lyrics, and even add expressive vocal parameters like breathiness, vibrato, and dynamics. However, when I attempted to create my first song, I encountered an issue: after clicking "play," nothing happened. After troubleshooting, I discovered that my voicebank model had been improperly converted to the ONNX format, meaning that it was incompatible with OpenUTAU. Users on the Discord server believed this to be a Mac issue, given that DiffSinger has recently had numerous problems running on Mac.

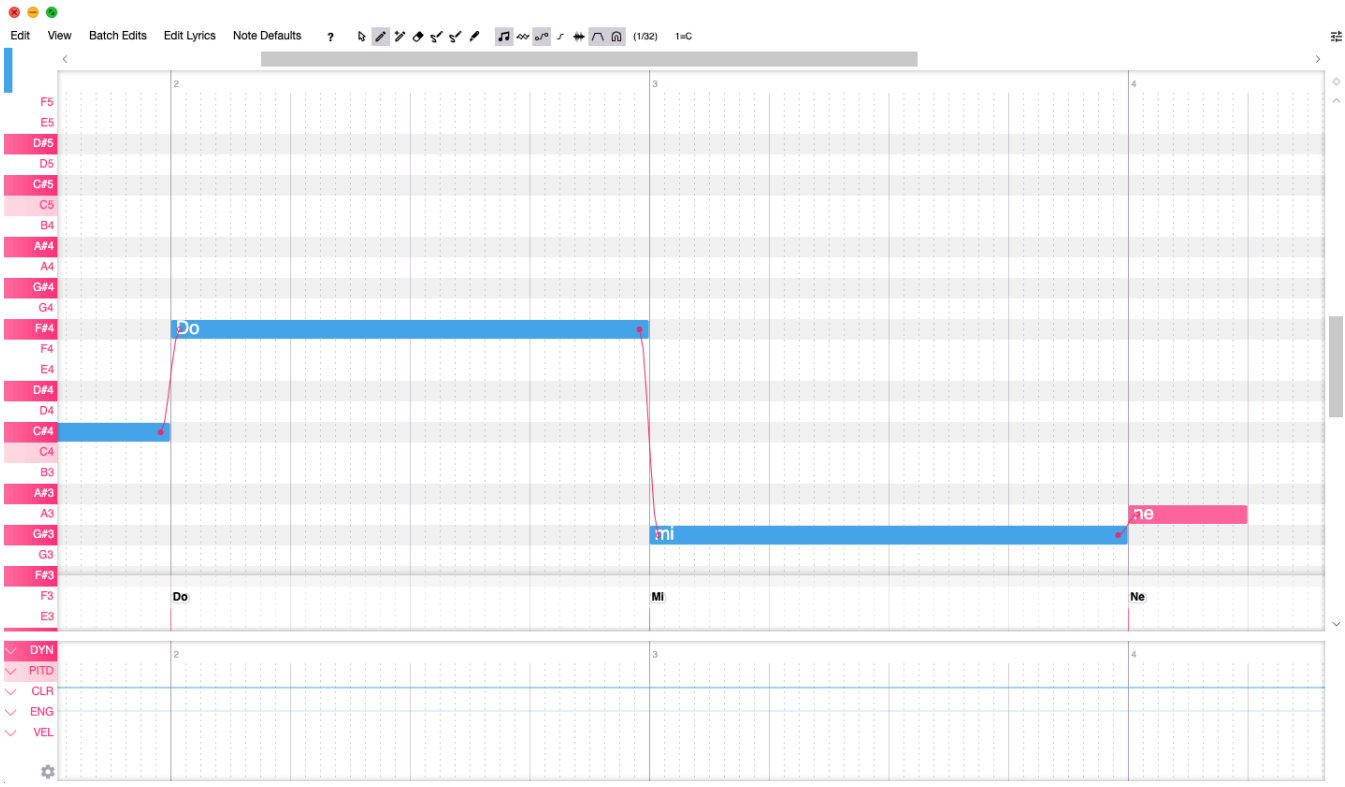

Figure 2. OpenUTAU interface

Results

Unfortunately, I was unable to repeat the process on a Windows machine in time to use my own voicebank. However, I did create a song using “Tiger”, a voicebank that I found on Discord that was created by @spicytigermeat on GitHub. The results are less than ideal, in part

because “Tiger” was trained on English pop music, a far cry from medieval Latin chants. I also added a chorus effect and a preset cathedral reverb in Ableton in an effort to bring the aesthetic closer to that of medieval chants. However, the output still sounds like heavily autotuned English, lacks musical direction, and has unnatural transitions between notes. On the positive side, the diction is clear, and the consonants are well-pronounced, which is a clear improvement over the previous Cantus Ultimus playback feature.

Figure 3. “Tiger” singing *Ecce dies venient* from Einsiedeln, CH-E 611

While the results of the project are certainly disappointing, I was able to lay the groundwork for future development despite the many technical issues in labeling, encoding, and training. Moving forward, I plan to continue building on the groundwork outlined in this project and will redo the training on a Windows computer, which will hopefully allow me to use my own voice to synthesize audio for medieval Latin chants and integrate the feature into Cantus Ultimus.

Bibliography

[1] Cantus: A Database for Latin Ecclesiastical Chant. Directed by Debra Lacoste (2011-present), Terence Bailey (1997-2010), and Ruth Steiner (1987-1996); developed for the web by Jan Koláček (2011-2023), McGill University Distributed Digital Music Archives & Libraries Lab - DDMAL (2023-present); and funded through the Digital Analysis of Chant Transmission project at Dalhousie University, Halifax, Nova Scotia, Canada (SSHRC 895-2023-1002), https://cantusdatabase.org.

[2] Liu, Jinglin, Chengxi Li, Yi Ren, Feiyang Chen, and Zhou Zhao. "Diffsinger: Singing voice synthesis via shallow diffusion mechanism." In Proceedings of the *AAAI conference on artificial intelligence*, vol. 36, no. 10, pp. 11020-11028. 2022.