Computational Creativity Final Project
Singing Synthesizer

Sara Adkins, 210077786

School of Electronic Engineering and Computer Science
Queen Mary University of London, UK
s.adkins@se21.qmul.ac.uk

1 Introduction

For my final project, I developed a singing synthesizer that generates lyrics and a melody, then synthesizes a woman singing them. The system is made up of 3 stages. First a GRU model generates lyrics, then an LSTM-GAN generates a matching melody line. Finally in stage 3 a DDSP autoencoder synthesizes the voice singing the melody and lyrics. The system in presented in a Google Colab notebook that allows the user to adjust model parameters and select their favorite outputs from each stage.

2 Background

Timbre transfer is a popular example of Google's Differentiable Digital Signal Processing(DDSP) library [3]. Their autoencoder model is able to make a sung melody sound like it was produced by another instrument such as a violin or a saxophone, but this concept of timbre transfer can't be applied the other way around due to missing lyric information. For my project, I set out to alter the DDSP framework to encode phoneme labels in addition to pitch and loudness to enable the autoencoder to synthesize sung lyrics given a time-aligned melody line.

The idea to embed phoneme information into a voice synthesis model was presented in the XiaoiceSing voice synthesis system[4]. A phoneme is the smallest unit of speech, so the combination of pitch and phoneme information is useful for singing synthesis. XiaoiceSing transforms an aligned sequence of phonemes and MIDI pitches into embeddings, which are used as features to the synthesis model. Their example results are impressive, and could easily be confused with a real human singer. However, my goal was to generate a more unique singing output; one where the lyrics could be understood but didn't sound exactly human. I was inspired by Holly Herndon's Proto album, which features the output of a SampleRNN model trained on her singing voice. Herndon's AI system isn't a very talented singer by human standards, but the outputs are far more interesting and surprising. I wanted my vocal synthesizer to behave in a similar way, focusing on generating interesting approximations of human singing rather than modeling a human singer perfectly.

Using the DDSP library to generate a singing voice was explored in Alonso 2021[1], but the results were disappointing. The examples provided sound vaguely speech-like, but it is impossible to decipher any syllables in the audio output. This may be because their implementation did not directly encode any lyric information during training or test, they relied on the latent vector representation to store vocal information. The DDSP speech synthesis results can likely be improved by conditioning on phonemes directly rather than relying on the latent vector.

3 System Description

My singing synthesis system consists of three parts: lyric generation, melody generation and voice synthesis. At each stage in the process, the user can adjust the predictability of the model and choose from a sample of generated outputs before moving on to the next stage.

3.1 Lyric Generation

To generate lyrics for the synthesizer, I implemented a 2-layer GRU network that generates a sequence of characters given a primer word or phase. I trained the network on character embeddings extracted from the Song Lyrics Dataset[6], and used [5] as a starting point for my code. I decided to extract only female artists from the dataset, to shape the identity of the voice AI system as a female singer.

The user is able to interact with the lyrics generator by setting the output length and primer phrase. They can also control how closely the outputs match the training distribution by adjusting the temperature parameter as a metric of predictability. The system generates 5 lyric outputs for the user, letting them choose which one should advance to melody generation.

3.2 Melody Generation

I used the pretrained LSTM-GAN model presented in Yu 2019[7] to generate a MIDI melody that aligns to the selected lyrics. The generator model, consisting of a 2 LSTM layers, takes a lyric embedding as input to generate a corrresponding MIDI sequence. The discriminator model, also a 2 layer LSTM, attempts to differentiate a real MIDI alignment from a generated one.

The user is again able to select a temperature level to control the predictability of the model, and they can also choose to overwrite the generated lyrics with their own. 5 melody options for the lyrics are presented to the user, and they can choose which melody to synthesize in stage 3.

3.3 Voice Synthesis

This final stage of the system runs the selected lyrics and melody through a DDSP model to generate the synthesized voice. DDSP uses an autoencoder architecture where the encoder stores pitch, loudness and a general latent vector. The decoder outputs parameters for harmonic filterbank and filtered noise signal processing blocks, which are combined to produce the final audio output. I modified the architecture to encode phoneme information, as shown in figure 1.

To enable the model to synthesize a voice timbre, I trained my phoneme-enhanced harmonics+noise model on the Children's Song Dataset[2]. The CSD dataset consists of 50 English folk songs, sung by the same female singer with time-aligned syllable information. I used the Prosodic Python library to parse each syllable to a list of phonemes, then manually stretched the vowel phonemes of each syllable to fill the note duration while keeping consonants set at 50ms, approximating how humans sing. This manual adjustment was necessary because the dataset timestamps were at the syllable level and not the phoneme level.

The phoneme autoencoder is fed the audio from the generated melody and the aligned phoneme vector to synthesize the final singing output as shown in figure 1. Because an autoencoder is a determinsitic model, there are no user adjustable parameters at this stage.

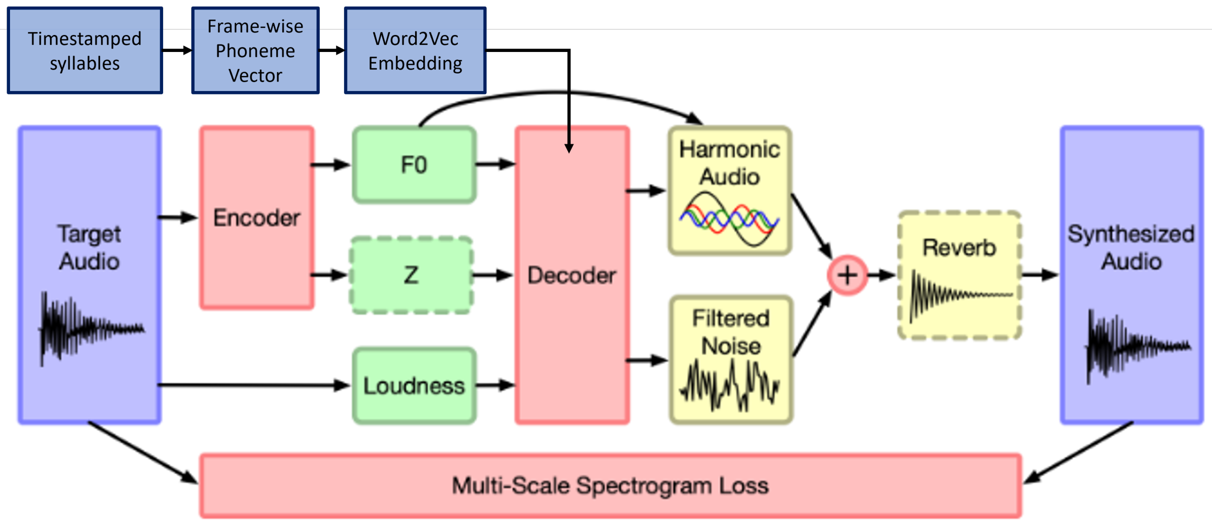

Fig. 1: DDSP harmonics and noise model, altered to encode phoneme information. Original graphic taken from https://magenta.tensorflow.org/ddsp

4 Experiments and Results

4.1 Lyrics Output

The quality of lyric generation depends on the temperature and primer. Generic primers like ”the” can produce nonsensical output as shown in row 4 of figure2. Using low temperature values can have the problematic consequence of copying a phrase from the training data. For example row 1 has copied lyrics from Taylor Swift’s Bad Blood. This also occurs in row 3 when the model amusingly states ”dinner you’re cold”, but then goes on the copy Katy Perry’s Hot and Cold for the rest of the output. Unique primers like ”ice cream” produced surprising yet coherent results. Row 5 nicely predicts ”truck” after ”ice cream” and then produces to rhyme ”truck” with ”luck.” We can also see in rows 6 and 7 the model finds 2 ways to correctly expand on the ”in my” primer. Overall, the system is able to produce coherent and interesting results after some experimentation with primers and temperature. A big downside is the model doesn’t form complete sentences, so the lyrics it generates do not feel like complete thoughts.

4.2 Melody Output

The melody generation system has minimal user control, but the predictability of the output can be adjusted with the temperature parameter. Temperature values above 0.25 produce random output that did not make for a nice melody, but lower temperature results are promising. Lower temperature values tend to have a smaller range and center the melody around a single note(see figure 3), mirroring the structure of real melodies. Higher temperatures have a wider pitch range and are not centered around a root(see figure 4).

I also found higher temperature melodies can insert unpleasant silence in the middle of the melody or

Temperature	Primer	Length	Output
0.05	”I said”	50	i said now we’ve got bad blood hey cause baby now
0.95	”I said”	50	i said keep it up guns whethehen your head to the heart
0.4	”dinner”	100	dinner you’re cold you’re yes then you’re no you’re in then you’re...
0.4	”the”	50	the catch down hahaha eheh eheh yeha yeha eheh eheh
0.4	”ice cream”	50	ice cream truck i try my luck and catch up but it’s always
0.4	”ice cream”	50	ice cream tree i’m gonna take you out take you out
0.7	”in my”	100	in my heart i had you find it all in my chest and it’s all on for the goodbye...
0.7	”in my”	100	in my head i cant hit a broke hoe i stand up from what we could be...

Fig. 2: Selected outputs from the lyric generation model

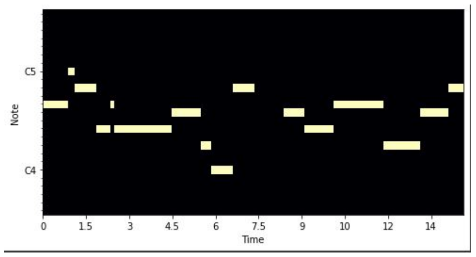

Fig. 3: Melody output for *in my heart i had you find it all in my chest and it’s all on for the goodbye i can’t lie i’m* at temperature **0.05**

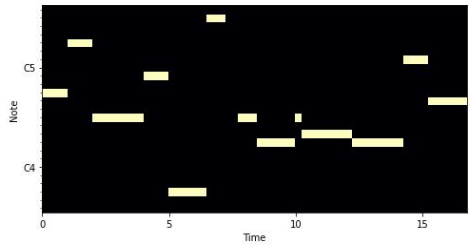

Fig. 4: Melody output for *”ice cream tree i’m gonna take you out take you out”* at temperature **0.20**

include too many short notes, as shown in 5. Overall, the melodies produced by the system are tonal and believable but bland. It is difficult to find a temperature parameter that produces surprising output without it just sounding atonal. The model does not enforce a meter, but at lower temperatures most outputs still have a sense of beat. The model also does not enforce ending on a cadence, so similar to the lyric generation system the outputs are left sounding incomplete.

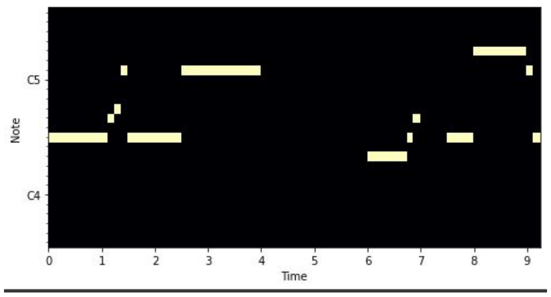

Fig. 5: Melody output for *”ice cream tree i’m gonna take you out take you out”* at temperature **0.20**

4.3 Overall Synthesized Output

The phoneme-conditioned DDSP model I developed did a great job at resynthesizing validation data from the Children's Song Dataset, but had performance issues generalizing to other sources. My initial plan was to synthesize the melody using a MIDI software instrument, but the loudness plots of the audio were very far off from the expected distribution and had artificial spikes at the note onsets(see figure 6). This caused noisy synthesized output that decay too quickly.

As a workaround, I generated artificial loudness curves for the melody output to more closely match the

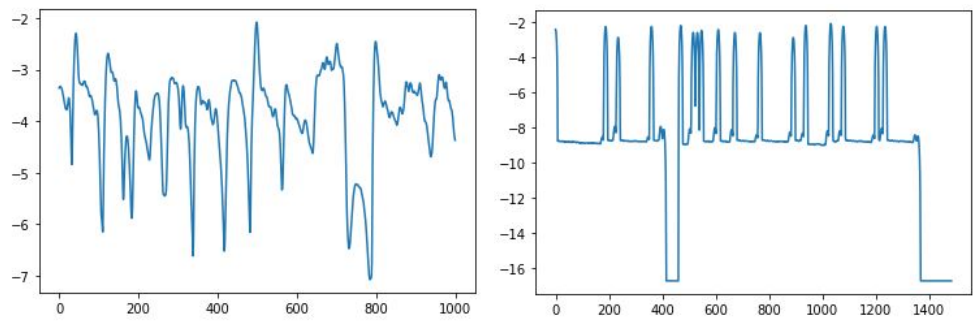

Fig. 6: Left: loudness plot from training data excerpt, Right: loudness plot from synthesized MIDI data

expected loudness distribution of the model. Figure 7 shows the extracted pitch and artificial loudness from a generated melody. To create the loudness curve, I set each syllable onset to a random value in the range [-2.0,-3.0]dB and each offset to a random value in the range [-4.5, -5.5]dB, interpolating between the values for each frame. These synthesized curves more closely match the training data distribution shown in 6 left.

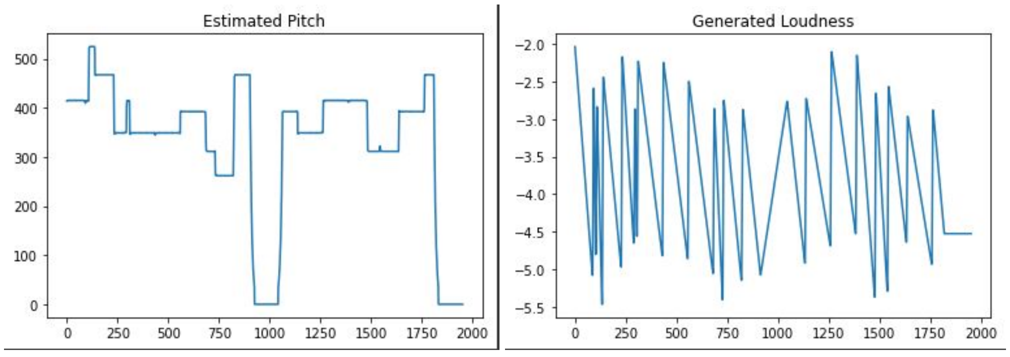

Fig. 7: Left: extracted pitch from generated melody, Right: artificial loudness from generated melody

5 Discussion

The outputs of the voice synthesis system were definitely speech-like, but not all of the syllables were clearly articulated. Longer held notes were easiest to understand, which is unsurprising given most of the training songs were at a slower tempo. While it is hard to make out the lyrics of the generated output, the system did succeed in creating a voice that is human-like but uniquely alien. Since the model was trained on a single vocalist, the system feels like it has its own identity as a singer. The voice is a mix of human woman and robot, sounding somewhat similar to a phase vocoder effect. There are also some interesting artifacts when the pitch estimation makes a mistake. The audio slides from the current frequency down to a very low one and back up for the next note, creating an unexpected but intriguing wailing effect.

The Colab notebook is intended as a casual creator tool for the singing synthesizer. The user does not need any musical background to generate the lyrics, melody or voice. I made the notebook interactive by by allowing the user to select from a small set of outputs at each stage and adjust temperature parameters. Presenting the user with a manageable set of options helps avoid choice paralysis, and avoid the frustration of having to continuously run the model to get a single output.

Despite the interactively, the system does still feel like a black box due to limited adjustment choices and a lack of explainability. While the user can adjust the ”expectness” of the outputs with temperature, it can be difficult to find a value that is surprising but still coherent with just this one knob. The model also does not provide any explanation for why it chose a melody for a lyrical phrase, and tends to gravitate towards major keys which do not work well for somber lyrics. This can cause the user to feel like they aren’t playing an active role in the creation of the final artifact.

6 Conclusions and Future Work

The voice synthesis system presented allows casual creators to create a sung melody and lyrics from nothing. The created audio clips could be used in a larger composition, and the consistent timbre of the synthesis voice gives it its own unique identity. This project has shown that it is possible to generate singing sounds from a DDSP model, but more work is required for the voice to be fully understandable. Future work on this front could explore different ways to generate a realisitic loudness curve for a MIDI melody, rather than relying on linear interpolation. The underlying model could also be improved by using a dataset with phoneme-level timestamps rather than inferring phoneme duration from syllables, however I have not found such a dataset.

The singing synthesizer could also be improved by adding output curation. Rather than having the user select from a random subset of outputs, a curator model could rate many outputs and present only the most promising options to the user. Finally, the system could be made more usable for composition ideas by adding in control for music parameters such as key, meter and cadences. With some more fine-tuning, this voice synthesizer could be used to create whole vocal sections for a song.

References

1. J. Alonso and C. Erkut, “Latent space explorations of singing voice synthesis using ddsp,” 3 2021. [Online]. Available: http://arxiv.org/abs/2103.07197

2. S. Choi, W. Kim, S. Park, S. Yong, and J. Nam, "Children's song dataset for singing voice research," 2020. [Online]. Available: https://program.ismir2020.net/static/lbd/ISMIR2020-LBD-435-abstract.pdf
3. J. Engel, L. Hantrakul, C. Gu, and A. Roberts, "Ddsp: Differentiable digital signal processing," 1 2020. [Online]. Available: http://arxiv.org/abs/2001.04643
4. P. Lu, J. Wu, J. Luan, X. Tan, and L. Zhou, "Xiaoicesing: A high-quality and integrated singing voice synthesis system," 6 2020. [Online]. Available: http://arxiv.org/abs/2006.06261
5. P. Paialunga, "Song lyrics generator," 2020. [Online]. Available: https://github.com/PieroPaialungaAI/SongLyricsGenerator
6. D. Shah, "Song lyrics dataset version 5," 2021. [Online]. Available: https://www.kaggle.com/datasets/deepshah16/song-lyrics-dataset
7. Y. Yu, A. Srivastava, and S. Canales, "Conditional lstm-gan for melody generation from lyrics," 8 2019. [Online]. Available: http://arxiv.org/abs/1908.05551http://dx.doi.org/10.1145/3424116