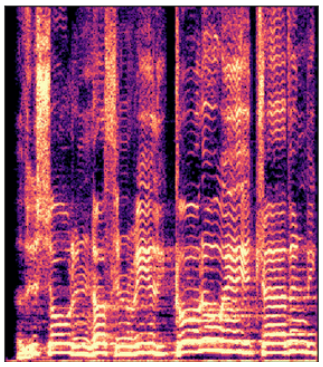

Using DDSP to learn and synthesize singing with adjustable latent space parameters

by
Harry Twigg 30748119

A document submitted in fulfillment of the requirements for the degree of
Aeronautics and Astronautics
at
University of Southampton

Abstract

In this thesis, a deep learning architecture called DDSP is used for synthesizing realistic vocal features. DDSP is a collection of machine learning models, differentiable versions of standard digital signal processing elements such as oscillators, noise filters, and other valuable tools for learning how to decode, learn and subsequently synthesize new musical audio signals.

The DDSP architecture is applied to a set of vocal samples of two different artists' voices to create a model for each respectively. The models extract pitch, amplitude, and timbre information from the vocal samples. Several inferencing tests are then conducted to determine the performances of the trained models at various tasks.

Finally, the results are concluded, and areas for improvement for future work are suggested.

Declaration

I, Harry Twigg, declare that this thesis and the work presented in it are my own and have been generated by me as the result of my original research. I confirm that:

1. This work was done wholly or mainly while in candidature for a degree at this University
2. Where any part of this thesis has previously been submitted for any other qualification at this University or any other institution, this has been clearly stated
3. Where I have consulted the published work of others, this is always clearly attributed
4. Where I have quoted from the work of others, the source is always given. Except for such quotations, this thesis is entirely my work
5. I have acknowledged all main sources of help
6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself
7. Where open source software has been used, I have abided by the license agreement that was provided with the software and made any derivative work publicly available under the same license
8. None of this work has been published before submission unless indicated.

Acknowledgements

I want to thank my supervisor, Professor Thomas Blumensath, for his support and guidance during my project. He let me go beyond the project's initial scope and into an area I honestly found very exciting whilst also keeping me on track with the project's original goals.

Supplementary Information

This thesis contains accompanying code and audio files not included in the thesis itself. It is recommended that listening to these is done while reading the experimental results.

These can be found in the following public GitHub repository:

https://github.com/harrytwigg/FEEG3003

Open source code licenses used in this code can be further found at the end of this paper in the section Open Source Licenses

Contents

List of Figures

1 Introduction

Using computerised methods to synthesise music programmatically has historically been a complex problem. Music contains thousands of sound features every second in the time domain that are difficult to teach accurately to a neural network.

Many potential applications would be opened up if a deep learning model could be devised to accurately learn, understand, and synthesise the fundamental features of music. These uses include many potentially artistic and business applications:

- Rapidly synthesising new vocal tracks for music production.
- Pitch transposing a piece of music, e.g. transposing a piece of music down an octave or up an octave.
- Changing room acoustics, e.g. if a piece of music was played in any echoic room, the model could be used to re-synthesise the same piece of music in an anechoic environment.
- Change the singing voice in a particular piece of music, similarly to deep fakes.
- Musical remixes of existing songs with different singers.
- Potentially brand new forms of artistic expression, with a neural network perhaps able to produce vocal features that are impossible to create naturally.

In recent years, deep learning methods have been applied to this problem, attempting to synthesise musical instrumentals and vocals. However, success to date has proven varied. Moreover, even if a network successfully learns how to interpret musical features, the output of models often sounds fake or jarring to the listener due to inaccurate pitch and timbre representations and a lack of temporal context.

Regardless, synthesising vocals has proven less successful than synthesising instrumentals; this is primarily due to the complexity of the human voice. The human voice consists of many different vocal modes and features that are difficult for a neural network to learn.

This thesis first explores various approaches to synthesising music using machine learning-based methods. Then, the merits and limitations of the approaches are evaluated regarding synthesising vocals and as a justification for the selection of DDSP. DDSP is a novel approach to synthesising music using deep learning.

DDSP is investigated with the ultimate goal of synthesising human singing with adjustable latent space parameters (F0 and loudness) and demonstrating its potential to be a viable solution to synthesising music.

1.1 Aim

This thesis aims to synthesize realistic human singing using DDSP with adjustable latent space parameters.

1.2 Objectives

1. Build a functioning machine learning model that can synthesize human singing with adjustable latent space parameters (F0 and Z).
2. Testing how accurate and adaptable the DDSP model is at synthesizing singing using several different inferencing tests, these validate that any created model behaves in line with the original DDSP model[7], and shows DDSP would be useful in future applications. They are as follows:
    - Almost identical resynthesis of the same song, using the same latent space parameters and using the same artist used in the model's training dataset
    - Independent control of the latent parameters loudness and fundamental frequency (F0)
    - Ensuring the words in model resynthesis is pronounced correctly (i.e. no babbling or other audible errors)
    - Ensuring the model can be trained in a reasonable amount of time and inferred in at least real-time
3. Derive constructive recommendations for future research based on past research and the results of this paper.

1.3 Methodology

1.3.1 Technical Requirements

A set of technical requirements were used to evaluate DDSP and other model architectures shown in this thesis. These are important for evaluating the utility of DDSP and other model architectures to achieve the programmatic and rapid vocal synthesis goals, as outlined in the Introduction.

- Use of teacher forcing or operator involvement in any methods. Teacher forcing leads to biases in the model outputs and limits the scalability and ease of using any derived models. Manually labelled data shall also be penalised similarly.
- Poor tonal quality in the output, e.g. it is noticeable that the model was generated digitally instead of recorded.
- Modular systems shall be evaluated positively because their elements can be built on separately, and the whole system acts less like a 'Black Box'.

- Any discarded information, e.g. phase that has been discarded during encoding (e.g. phase) that could be presented to the network shall also be penalised. It is hypothesised that this information could be used to improve the quality of the output.
- Model architectures specific to music and audio signal processing were preferred instead of more general ones. Furthermore, it was believed that directing the model towards specific musical features (such as sinusoidal harmonics) would be beneficial, rather than generalising to the entire audio signal.

1.3.2 Academic Requirements

Well cited papers or those in scientific journals were looked upon favourably, showing that other people have found the work valuable and, more importantly, credible. It was also desired that any researched papers have open-sourced code and that the code is available for use. Without this, the model cannot be quickly built without building the codebase from the ground up, which would take considerable time. Older methods that have not been built further were evaluated negatively, as this suggests that experts in the field have judged the work to be of no further benefit and hence obsolete.

1.3.3 DDSP Training

DDSP is fundamentally a modular and extendable deep learning architecture, enabling the time-wise modification of evaluated sound qualities called latents (pitch and loudness). Additionally, it features a series of differentiable versions of traditional signal processing techniques. After background research, DDSP appeared to be the most promising model architecture due to several factors discussed in Differentiable Digital Signal Processing.

Two different datasets were created, one for a male voice artist and one for a female voice artist, to see how the DDSP architecture would handle male and female voices differently.

For each artist, two different albums were picked of similar musical styles to ensure consistency of vocal style across the entire dataset; this was done to try and encourage the model to learn a specific timbre of voice.

The albums were picked so that they only had one voice on the vocal track to avoid any problems of the model mixing voices, any songs with cover artists or different singers to the leading voice were removed.

Each dataset was processed through a pre-trained model called Spleter[12]. This pre-trained model separated the vocal track from the instrumentals for each song in the album. Consequently, large datasets could be easily created featuring a singular vocal track and enabling datasets 10x the size of the paper this work was based on[1].

Two models were then trained using the DDSP library[3], one for each respective dataset. The code for these was derived from the DDSP Library and a variation of DDSP designed for singing[1]. Model hyperparameters were kept the same as in the Singing DDSP paper[1] as the researchers had

demonstrated thorough testing of which hyperparameters were the best. 200,000 epochs were used for each dataset; this was deemed sufficient for sufficient model quality whilst keeping the training time manageable.

1.3.4 DDSP Inferencing

Following the training of both models, the models underwent several inferencing tests designed to evaluate their encoding of the latent characteristics (pitch, loudness, and timbral transfer) and to evaluate the flexibility of the DDSP architecture in a variety of different situations.

1. The models were inferred on the same dataset used for training to test the model's accuracy in predicting frames from the training dataset.
2. A pitch transposition was attempted. Vocal samples from the original dataset were transposed up and down an octave to see how the model would perform on unseen vocal ranges.
3. A mono-pitch inference test was conducted to determine the model's ability to produce a single pitch.
4. A log-linear loudness inference was attempted to gauge the model's ability to modify the loudness of the vocal samples.
5. For the best model, timbral transfer tests were conducted. The best was determined by performance in previous inferencing tests. The timbral transfer tests were conducted on unseen male and female voices to determine performance with different unseen voice types. The test frames were separated similarly to the test datasets using Spleeter. In the tests, all latents remained unmodified to isolate the effect of changing the vocal artist.
6. Again, on the best model, an inference test on a track with instrumentals was conducted to observe the model's performance on unseen instrumentals within a track. Again, all latents remained unmodified to isolate any effect of the instrumentals. The track was the female voice artist from the timbral transfer test. The same timestamped frame was used to compare the output between the inferred frames for both tracks.

Finally, in light of the experimental results and other academic research, recommendations for future work in the field are made.

2 Literature Review

All the literature on vocal sound synthesis methods is evaluated here; they are laid out in roughly chronological order. DDSP is outlined in the greatest depth as it was the technique chosen for further investigation.

2.1 Symbolic Methods

Using computers to generate music is a vast field of research dating almost as far back as the invention of computers themselves[13]. However, the first research into using machine learning to generate music goes back to the 1980s[25][15] with the very first research on symbolic data. These higher-level representations often took the form of MIDI or other musical notation.

Symbolic based models provide a high-level abstraction of the musical piece, meaning that they are easier to train as the model does not have to worry about the physical process of producing sound. However, they are significantly limited to music described in MIDI notation, i.e. vocals and other instruments with unique modes of being played cannot be used.

> MIDI (Musical Instrument Digital Interface) is a protocol and digital interface for musical notation. It is widely used for recording, editing and playing music.[17].

More recent works in the field are applying modern transformer machine learning architectures to generate symbolic music with long-term structurework[14]; this is a challenge experienced in many music synthesis models. Transformer based architectures use what is called an attention-based mechanism to generate long term structure[26]. Attention-based mechanisms are different to convolutional or recurrent neural networks, and they can be used with or without them.

2.1.1 Critical Evaluation

Due to their high level of abstraction, symbolic methods generally have small models with few parameters and are easy to train. As a result, they prove helpful in generating abstract long term musical structures. Furthermore, the relatively recent papers showing long term structure[14] and

the use of attention based mechanisms[26] show that symbolic methods have future potential and relevance.

Nevertheless, symbolic methods have low levels of resolution and can only be used to generate music described in terms of midi or similar notation. This limited expression is a limitation of symbolic methods, as they cannot be used to generate raw time-series audio features. Furthermore, they have no way of generating the intricacies and expression of the human voice; this is hard to describe symbolically.

Sadly the disadvantages of symbolic methods (specifically their lack of output expression) outweigh any benefits they offer in forms of long term musical structure. Nevertheless, even if they are limited in their range of expression, they could be helpful if combined with another model capable of synthesizing time-series audio data. The symbolic model could provide the other model with a long term musical structure, which uses this to generate local time-series audio.

2.2 Raw Waveform Based Encoding

A subset of deep learning research has focused on waveform-based methods and encoding music audio directly as raw time-series audio data.

> Deep Neural Networks (DNNs) are a type of neural network with multiple hidden layers between the input and output layers. They are well suited for learning complex functions humans or other artificial neural networks cannot intuitively see from large datasets.

There are many potential benefits to crunching low-level audio data. Firstly, information discarded from spectrograms (a representation discussed later in 2.3), e.g. phase, is not lost. Secondly, using raw audio does not limit the number and depth of features that can be learned, meaning such a model could potentially learn the intricate features of the human voice.

One of the problems of waveform based models is their lack of long term structure; this is caused by the sampling period being so short and the many thousands of samples that make up one continuous raw audio track[4].

2.2.1 Jukebox

The most influential model employing waveforms for music sound synthesis is called Jukebox[5]. The Jukebox model is a deep neural network that learns to reconstruct music's vocal (and instrumental features) features from the raw audio data.

What is significant about the paper is its method of attempting to overcome the lack of long term structure. An autoencoder compresses input raw audio at the Nyquist Frequency to a discrete space using a technique called Vector-Quantized Variational Autoencoders (VQ-VAE)[5].

> The Nyquist frequency is the highest frequency that can be represented by a sampling rate of an encoded audio signal so that the original signal can be reconstructed[11].

Several independent VQ-VAE levels are used, retaining different levels of resolution up to 128x encoding. Lower levels capture local music structures, e.g. timbre and local pitch, whereas higher levels capture higher-level long-range music structure features.

Each level is then trained using a sparse transformer-based model to learn the probability distribution of the VQ-VAE at each level of resolution. Finally, each VQ-VAE code is conditioned using the higher-level code during upsampling. Doing this enables the reconstruction to use all the VQ-VAE levels.

The model can be conditioned on lyrics, genre, and artist during inference. Inferred music features singing, instrumentals and some resemblance to a long-term structure. Music vocals (and language) are also synthesised and sung by the model on its own accord.

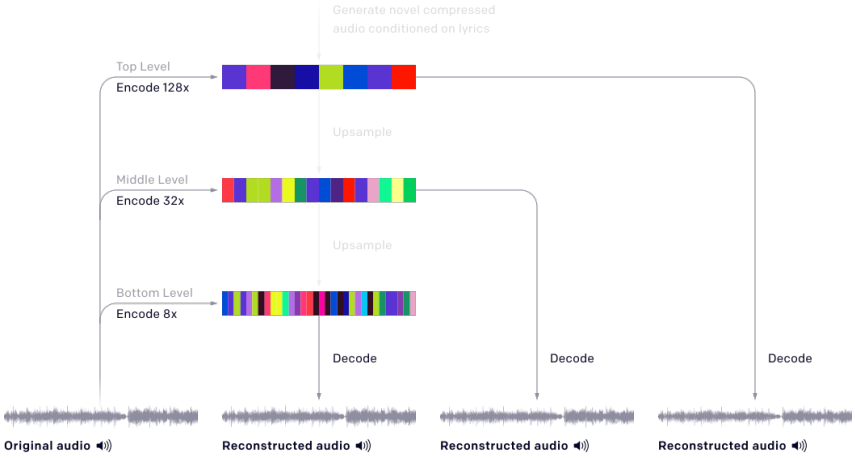

Figure 2.1: VQ-VAE Encoding and Compression: Successive levels further compress the raw audio data, discarding irrelevant information[5]

2.2.2 Critical Evaluation

Although local musical coherence and timbre are good in Jukebox, the longer-term musical structure is not fully present. The upsampling process also introduces significant noise into the final audio, which sounds jarring to the listener. Another significant disadvantage is the sheer size of the model, caused by the high volume of low-level time-series data). This large size means it takes multiple hours to infer one minute of music. This slowness limits its broad applicability as it is prohibitively expensive and prevents real-time applications, two significant issues to commerciality.

The model also functions much like a black box, preventing us from gaining any critical information about how it is synthesising its audio. This further limits its potential uses as we cannot independently modify model parameters such as the pitch and timbre of generated music. Finally, waveform based models require significant training data to train the model and extract relevant musical features accurately.

Sadly the lack of real-time features and prohibitively large model size limit Jubebox's potential applications. Any real-world application needs to be real-time or near-realtime.

2.3 Fourier Methods and Spectrograms

Spectrograms have a long history but were first used with machine learning models to synthesise music at the start of the 21st century[19].

A spectrogram is a graphical plot of the decomposition of sound using the Short-Time Fourier Transform (STFT). It consists of 2 plots frequency against time and phase against time. Each point of the plots is coloured in amplitude/intensity of the decomposed audio signal at a specific point in time. The STFT is a sum of overlapped Fast Fourier Transforms (FFTs) of the audio signal. After calculation of the STFT, the sample is divided into overlapping frames of equal length, known as the hamming window. Finally, a Fast Fourier Transform is applied to each audio signal frame.

The outputted complex functions from the Fast Fourier Transform give spectrograms of amplitude and phase of different frequencies at each timestep.

2.3.1 Mel-Spectrograms

Mel Spectrograms are an adaptation of the spectrograms more suited to sounds intended to be heard by humans. Mel Spectrograms have amplitude/sound intensity adjusted along a logarithmic scale such that graphical distances between frequencies sound the same distance as human hearing would detect them to be. This adjustment enables machine learning models to learn how to produce audio sequences that sound more natural to a human listener.

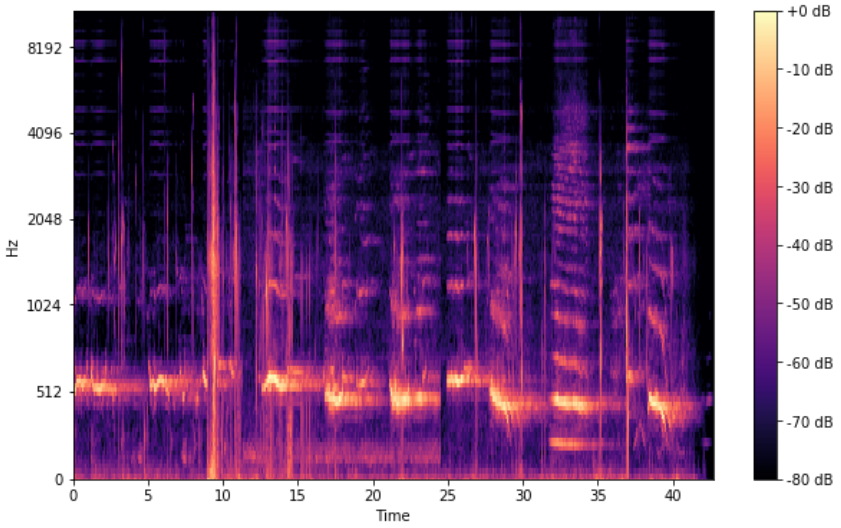

Figure 2.2: An example of a Mel Spectrogram showing the amplitude of different frequencies in a sound over time[10]

There is no standardised formula for converting frequncy onto the mel logarhthmic scale as it is up to interpretation the adjustment level that is required for human hearing. However, the most common formula is[20]:

$$m = 2596log(1 + \frac{f}{700}) = 1127ln(1 + \frac{f}{700}) \tag{2.1}$$

2.3.2 Intepreting Phase

The STFT is a complex function; however, only the magnitude part is currently utilised by most models, and the phase part is discarded. Therefore, any reconstructed signal cannot have the same phase as the original signal if the phase information is lost. Furthermore, human hearing can distinguish between two sounds with the same frequency but different phases, meaning that sounds synthesised without phase can sound unnatural and different to the original, regardless of the frequency.

Success at interpreting the phase spectrogram to date has been limited. As a result, most models discard the phase part of the signal and make models purely off the frequency spectrogram. Discarding the phase is a problem as the phase is a crucial part of spectrogram representation. The phase spectrogram makes the spectrogram image representation fully convertible back to audio, though it is challenging to work with for two main reasons.

Firstly, the phase spectrogram appears random, making it challenging to distinguish meaningful information from noise.

Secondly, the spectrogram phase is a cyclic quality. Cyclic qualities are more challenging to interpret than non-cyclic qualities as they are not continuous. One paper called GANSynth[6]

overcomes this problem by calculating the phase difference between individual timesteps of a spectrogram and making 2π adjustments for when the phase wraps around. This phase difference is called the instantaneous frequency and provides far more informative information than the raw phase. Instantaneous frequency over harmonics, for instance, is expected to be constant.

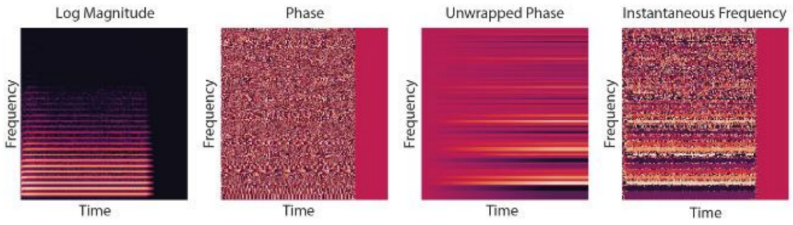

Figure 2.3: Unwrapping Spectrogram Phase: An illustration from the GANSynth paper of how the phase of a spectrogram is adjusted to make it more interpretable to a neural network[6]

2.3.3 Spectrogram Evaluation

Spectrogram based encoding for music sound synthesis is currently the best method for encoding musical sounds due to their ease of use and the low-level control over signal information that does not hinder their interpretation (unlike raw waveform encoding). Conventional image processing models, e.g. Convolutional Neural Networks or Recurrent Neural Networks, can be used to process the spectrogram. These models can extract local sounds at specific frequencies from a spectrogram and learn how to reproduce them. This extraction is possible due to the separated frequency and timewise position of sounds that form specific image patterns that an autoencoder can be trained to recognise. An example in this case of singing is shown in the following Figure 2.4:

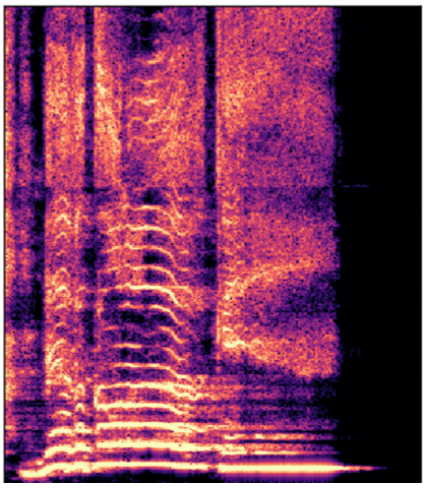

Figure 2.4: Spectrogram Features: Regular harmonics present in singing can be observed in the bright horizontal regions of the spectrogram that repeat at regular frequency intervals. A model could be trained to extract these features and learn to resynthesize them

Additionally, one of the spectrogram's defining advantages is that it can be used to reconstruct the original audio signal from the inverse Fourier Transform. The reconstructed audio signal will be almost identical to the original (phase-matched) if phase and magnitude spectrograms are used. Spectrograms are at the core of many recently published music synthesis models and papers, showing academic relevance.

However, most models based on spectrograms, e.g. raw CNN-based models, are limited and do not make use of biases in sound, for instance, the tendency of natural sounds to oscillate sinusoidally at harmonic frequencies according to the harmonic plus noise model. In addition, many traditional autoencoder-based RNN models do not enable the configuration of specific sound features, e.g. fundamental frequency or loudness. Additionally, spectrogram based models are not without problems (e.g. the discarding phase of information).

Therefore, although helpful, spectrograms cannot be used to synthesise audio signals on their own. To achieve widespread academic and commercial use, a different representation to help a model relate them to sound is required, for example, Differentiable Digital Signal Processing.

2.4 Differentiable Digital Signal Processing

Differentiable Digital Signal Processors (DDSP) are differentiable versions of traditional digital signal processing elements (such as harmonic oscillators and filtered noise banks) that can be integrated into machine learning models[7]. They are a relatively novel invention but build on the foundations of traditional spectral encoding and modelling techniques such as Mel-Spectrograms and Fourier Transforms. Although they were designed to model musical instruments, they can be extended to the human voice.

The DDSP network is modular and consists of multiple separate feedforward components (featuring a separate encoder and decoder) instead of a single recurrent based autoencoder network. Information is fed between the encoder and decoder in time-dependent information called latents; these are fundamental frequency and loudness, and Z (the hidden outputs of the encoder).

The model aims to take advantage of what the authors call inductive biases of sound instead of making the model figure out all the features of sound itself. This idea is well validated and is known as the harmonic plus noise model of spectral modelling synthesis[24]. For example, an instrument's track could have harmonic vibrations at multiples of the fundamental frequency (modelled by the harmonic oscillator) and noise components (modelled by the filtered noise bank). Noise is derived from passing white noise through a series of filter banks that change over time).

2.4.1 Spectral Modelling Synthesis

DDSP synthesiser elements are based on traditional synthesiser-based components that can be combined with an encoder and decoder to form a complete machine learning model. DDSP uses a type of sound modelling called Spectral Modelling Synthesis[22] to model sound, with the components being modified, so they are differentiable. By recreating them as differentiable components, the gradients of each block can be accessed in the machine learning model, enabling them to be configured by the model. Synthesisers take the network outputs of the decoder and use them to synthesise an output audio signal.

Harmonic Oscilator

The harmonic oscillator is the first of the 2 Spectral Modelling Synthesis components. It consists of a bank of oscillators that output a sinusoidal signal denoted as $x(n)$ where n represents discrete time steps and is equal to the sum of the sinusoidal waves. The harmonic oscillator can be expressed as:

$$x(n) = \sum_{k=1}^{N} A_k(n) \sin(\phi_k(n)) \tag{2.2}$$

- $x(n)$ is the output signal
- $A_k(n)$ is the amplitude of the kth sinusoidal oscillator
- $\phi_k(n)$ is the instantaneous phase of the kth sinusoidal oscillator

The phase at a certain point in the output signal can be calculated as follows using the instantaneous frequency:

$$\phi(n) = 2\pi \sum_{m=0}^{N} f_k(m) + \phi_{0,k} \tag{2.3}$$

- $\phi_k(n)$ is the instantaneous phase of the kth sinusoidal oscillator

- $f_k(m)$ is the instantaneous frequency of the kth sinusoidal oscillator
- $\phi_0(k)$ is the initial phase of the kth sinusoidal oscillator

Like actual sound, each harmonic oscillator's frequency $f_k(m)$ is an integer multiple of the fundamental frequency $f_0(n)$. For example the kth harmonic oscillator, the instantaneous frequency is defined as:

$$f_k(m) = k \times f_0(n) \tag{2.4}$$

The highest harmonic should be at the Nyquist frequency, which is half the sample rate or defined mathematically as:

$$f_{Nyquist} = N \times f_0(n) \tag{2.5}$$

Defining the highest harmonic in this way ensures that the harmonic oscillator can represent all of the possible signal ranges.

The initial phase $\phi_0(k)$ can be random, fixed or learned[7].

DDSP employs bilinear interpolation upon the fundamental frequency to increase the sample rate from the neural network and further smoothing to prevent artefacts. Bilinear interpolation is necessary as the neural network is trained at a lower sample rate than the original signal, whereas oscillators operate at the original sample rate.

> Bilinear interpolation is a mathematical technique enabling the estimation of new values between existing discrete values that are the functions of two variables. In DDSP, it is used to increase the neural network's sample rate to that of the underlying audio signal.

Filtered Noise

The filtered noise is a DDSP synthesiser component that uses a time-varying filter bank. In this, white noise is filtered by a Linear Time Invariant Finite Impulse Response Filter (LTI-FIR) to produce a noise signal. LTV-FIR filters are a type of filter that can vary over time, enabling different noise patterns to be generated at different time steps in the output signal.

Finite impulse response filters consist of mathematical impulse responses, meaning that the filter is of finite duration, enabling the filter to change and, over time, model different stochastic noises in the output signal.

As the filters are linear time-invariant, the noise filter can be characterised by its response to different frequencies of sinusoids, known as its frequency response. The filter is also independent of time.

It is worth noting that the noise filter is applied in the frequency domain; this is done to avoid phase distortion. Phase distortion occurs when a filter's phase response is not linear in the frequency domain. A lack of linearity in the filter domain can distort the filter's output, making synthesised sounds sound unrealistic.

The neural network was tasked with predicting frequency domain transfer functions of the filter for every frame of the output signal, denoted as H_l[7].

For all notations with subscript l, l denotes that the values are for the lth frame of the output signal.

H_l is in the frequency domain; it shows the filter's frequency response. Even though LTV-FIR filters are not linear time-variant, the network can vary the FIR filter over time, allowing its output to change over time and permitting different noise patterns to be modelled.

As linear time-invariant filters can be denoted by their impulse response, the frequency response of the filter can be denoted by its impulse response h, as shown below:

$$H_l = \mathscr{F}^{-1}(h_l) \tag{2.6}$$

- H_l is the frequency response of the filter in the frequency domain
- h is the impulse response of the filter
- $\mathscr{F}^{-1}$ is the inverse descrete Fourier transform

In order to apply the time-varying filter to map the input to the output, the following is done:

1. An input white noise signal is split into non-overlapping frames of fixed hop size, each one to go with a certain impulse response h_l.
2. Frames are then multiplied in the fourier domain:
   - $Y_l = H_l X_l$
   - Y_l is the output signal, where $X_l = \mathscr{F}(x_l)$, where $\mathscr{F}$ is the Discrete Fourier Transform (DFT)
   - X_l is the input signal, where $Y_l = \mathscr{F}^{-1}(y_l)$, where $\mathscr{F}^{-1}$ is the Inverse Discrete Fourier Transform (IDFT)
3. Output audio frames are then retrieved using the inverse Fourier transform: $y_l = \mathscr{F}^{-1}(Y_l)$
4. Output audio frames are then comnbined, using the same hop size as the input audio frames.

> Hop size refers to the number of samples between each frame; the number of samples in a frame equals the hop size.

The filtered noise synthesiser can generate many different noise patterns that are not harmonic and continuous. The original DDSP authors used them to generate unique instrument features,

such as the plucking of a violin, but they can also be used to generate consonants in the human voice [1].

2.4.2 Model Setup

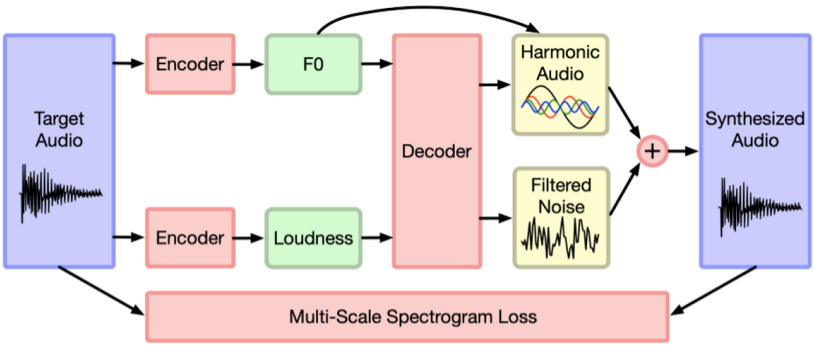

Figure 2.5: The DDSP Model Architecture: The model setup courtesy of a DDSP paper[1]. Standardised machine learning components are shown in red, the latent variables in green and the synthesizers in yellow. Note the feed-forwarding of F0 bypassing the decoder to directly control the harmonic oscillators.

The harmonic oscillators and filtered noise are combined to produce the synthesised output signal in what is known as the Harmonic plus Noise model. The combined output can also be passed through additional modules to produce different types of sounds, for example, a reverberation module to accurately model the acoustic characteristics of a string instrument; this is, however, beyond the scope of this paper. Therefore, no reverb or additional modules are included in this thesis.

The initial DDSP model[7] employed a modified autoencoder decoder setup where the autoencoder was trained to minimise reconstruction loss in an outputted synthesised audio. Training data is fed into the encoder in the form of Mel-spectrogram images. The autoencoder attempts to extract information from the input signal. F0 is directly fed into the harmonic oscillators as making them learn to reproduce this would be unnecessary.

A further latent quality called Z is extracted from the output of the encoder representing the outputs of the encoder; this cannot be modified. Instead, it is fed directly into the decoder. The original DDSP paper demonstrated that timbral information was encoded in this latent Z, enabling timbral transfer between various musical instruments to be possible.

Loudness and fundamental frequency, and Z are time-dependent, they can be denoted as such: $f(t), l(t), z(t)$.

F0, the fundamental frequency, can be extracted using an encoder in the standard model. The DDSP architecture is, however, very flexible. Therefore, different variations of this autoencoder

setup can be used. For example, instead of forcing the encoder to map F0, CREPE can be used to extract the F0 from the input signal.

> CREPE is a highly accurate pitch detection based Convolutional Neural Network (CNN)[16]. It can accurately predict pitch with an accuracy of over 99.9%. In addition, it is a pre-trained model that can be used with the DDSP library to extract the pitch from the input signal accurately.

The decoder then uses the F0, loudness, and Z to derive control values for the filtered noise and oscillator synthesisers, learning how to re-synthesise the target-audio track.

The fundamental frequency is, however, additionally fed directly into the synthesisers as it enables the model to respond to frequencies unseen during training[1].

Another benefit is that certain features can be explicitly controlled due to the modular design, for example, room acoustics. Some networks would implicitly pick up room acoustics. However, this method may introduce unnecessary mode covering, increasing potential training time. The DDSP model explicitly defines room acoustics using a reverberation synthesiser (see Figure 2.5). However, this is not used in this paper.

2.4.3 Measure of Loss

In order to train the model, it is necessary for a measure of loss to be defined. The autoencoder is tasked with minimising the reconstruction loss; this is the difference between the synthesised audio and the target audio. Unlike in a conventional autoencoder model, the loss cannot be defined pointwise as two waveforms may sound the same but have different pointwise characteristics. The loss function is therefore defined using a method called multi-scale spectral loss. The loss is defined as the sum of L1 differences and the L1 log difference between the target and synthesised magnitude spectrograms.

> L1 loss is defined as the sum of the absolute differences[27], the sum of the absolute differences between the magnitude spectrograms.

$$L_i = ||S_i - \hat{S}_i||_1 + ||\log S_i - \log \hat{S}_i||_1 \tag{2.7}$$

- L_i is the model loss FFT of size i
- S_i is the target spectrogram with a given FFT of size i
- $\hat{S}_i$ is the synthesized spectrogram with a given FFT of size i
- $||S_i - \hat{S}_i||_1$ is the L1 difference between the target and synthesized spectrograms
- $||\log S_i - \log \hat{S}_i||_1$ is the L1 difference between the target and synthesized logarithmic Mel-spectrograms

The total loss is then equal to the sum of spectral losses for different FFT sizes:

$$L = \sum_{i=1} L_i \tag{2.8}$$

Where $i \in (4096, 2048, 1024, 512, 256, 128, 64)$

2.4.4 Speech Synthesis Using DDSP

Further research on top of DDSP relating to vocals has been done for speech synthesis, although not directly designed for music; the paper introduces a slightly different system that modifies the DDSP architecture to more closely match the human voice[8]. Instead of using an autoencoder, the network uses a series of convolutional neural networks.

The method yields highly accurate results (although it is still noticeable that the output has been synthesised). Timbre was accurately measured, though consonants still sounded off. Unfortunately, the code used for the model was not available for download, limiting the value of the paper as its findings could not be replicated. It could, in theory, be reversed engineered though this would take considerable time and effort and is, as such, beyond the scope of this project. It is also possible that the results were cherry-picked from the original as only a limited number of samples were available.

2.4.5 Singing Voice Synthesis Using DDSP

A new team has conducted attractive work building on top of the initial paper to produce singing voice synthesis[1] specifically. This paper outlines how the DDSP model can be further altered to interpret better and learn how to model the human voice.

The model trained in this paper was hoped to model consonants using the filtered noise bank and vowels using the harmonic oscillator module; this is the same approach used later in this thesis.

The adaptation involved adding Mel Frequency Cepstral Coefficients (MFCC) an additional time-varying form of encoding. The coefficients together make up a Mel Frequency Cepstrum (MFC). A Mel-frequency cepstrum (MFC) represents the spectrum of a signal using a non-linear Mel frequency scale.

MFFCs have a long history of use in telecommunications and speech processing[9]. Thus, the original authors hypothesised that this representation would more accurately model the non-linearity of the human voice, thus making a vocal synthesis model more accurate.

The modified decoder and MLP can be seen in Figure 2.6. Additional MLP layers have been added to the decoder to allow for MFCs to be learned. Besides this addition, the DDSP architecture is the same as the original, relying on the harmonic plus noise model.

Multilayer Perceptron (MLP) is a feedforward neural network module. MLPs feature a directed graph, meaning that the signal path only goes one way. A non-linear activation function also characterises each node. Here the MLP is used to learn the Mel Frequency Cepstrum and its coefficients.

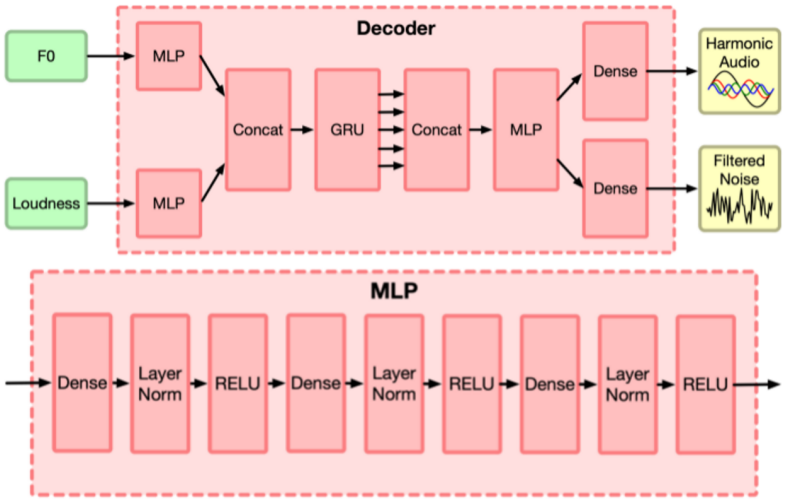

Figure 2.6: Modified DDSP Decoder and MLP[1]

Even on a small dataset, promising results were obtained. Timbre transfer was possible, and pitch and loudness were also accurately modelled. Furthermore, the performance of the modified decoder exceeded that of the original decoder.

Unfortunately, when the model was forced to recreate unseen sung lyrics, it was unintelligible, appearing to make stuttering noises. The model managed to obtain the correct loudness and pitch of the sound (except for the occasional pitch artefacts) but had no understanding of phonetics (i.e. the specific words that makeup speech). The paper suggests several steps for further work:

- Phonetic condition of the model to model the nuances of human singing and to model the lyrics.
- Using synthesisers more suitable for modelling the human voice.
- Pre-processing of the fundamental frequency to remove pitch artefacts.

2.4.6 DDSP Evaluation

In the original paper, the DDSP decoder quickly learned to re-synthesise datasets for a single instrument that sounded like the original audio sample.

A big pro of the method was that proper training on an instrument could be undertaken with as little as 15 minutes of training audio. Such a small dataset contrasts with models such as Jukebox, which require many hours of training audio as they are far larger models.

Due to the smaller model size, the model can operate with reduced training time and cost, making it more suitable for use in real-world applications and perhaps real-time applications.

Another plus was the interpretable and modular design of the model; individual factors such as pitch or loudness could be varied whilst keeping the other characteristics constant. This interpretation was possible because the model was conditioned to use pitch and loudness to re-synthesise the audio.

Furthermore, individual components can be changed, perhaps without rebuilding the whole model. E.g. the spectrum based encoder can be replaced with a midi sourced encoder, with the rest of the model architecture remaining unchanged.

It was eventually decided that using the DDSP model would be picked for further research due to its modularity, accuracy and ease of use.

The variation of DDSP designed for Singing Voice Synthesis Using DDSP was chosen as this is a relatively novel area of research and held the most exciting applicability. However, this has historically proven to be a complex problem to solve. The human voice is not a simple harmonic series with a constant timbre.

3 An Investigation in Using DDSP to Learn and Synthesize Vocal Features

For this project, the modified version of the DDSP archtecture based off *Latent Space Explorations of Singing Voice Synthesis using DDSP*[1] was used. The architecture differs from the original in that there are additional MLP layers.

All research was undertaken using Google Colab notebooks and cloud hardware, primarily NVIDIA Tesla V100 GPUs.

Two separate models were trained, one for a male voice (Chris Martin from Coldplay) and one for a female voice (Taylor Swift); this was done to see how the DDSP architecture would handle different voices, e.g. Alto or Tennor, differently. The songs for each can be seen in Figure 3.1.

Coldplay		Taylor Swift
A Rush of Blood To The Head		**Red**
Politik		State of Grace
In my Place		Red
God Put a Smile upon your face		Treacherous
The Scientist		I Knew You Were Trouble
Clocks		All Too Well
Daylight		22
Green Eyes		I almost Do
Warning Sign		We Are Never Ever Getting Back Together
A whisper		Stay Stay Stay
A rush of blood to the head		The Last Time
		Holy Ground
		Sad Beautify Tragic
		The Lucky One
Ghost Stories		**Folklore**
Always in My Head		Willow
Magic		Champagne Problems
Ink		Gold Rush
True Love		Tis the damn season
Midnight		Tolerate It
Another's Arms		No body, o crime
Oceans		Happiness
A Sky Full of Stars		Dorothea
		Coney Island
		Cownboy like me
		Long story Short
		Marjorie
		Evermore

Figure 3.1: Songs and albums in the training datasets for each model

Preliminary investigations on smaller datasets yielded significant overfitting and poor performance when F0 or amplitude latents were modified during inference. Therefore, it was hypothesized that two albums would be the optimal size for any training dataset. Therefore, datasets of any greater size would be preferred. However, larger models would be slower to train.

3.1 Dataset Preparation

3.1.1 Source Separation

Following the selection of the two albums for each artist and removal of any songs where additional vocal artists to the main vocalist featured were removed, all songs were passed through the pre-trained Spleeter model[23][23]. This pre-trained model can carry out source separation of instrumental and music tracks. Source separation was essential as the training dataset must not contain any instrumentals. The model outputted two separate tracks for each song, one representing the instrumentals and one the vocals. The instrumental tracks were discarded.

Using Spleeter to split existing songs with instrumental components opened up the possibility of using a more significant number of songs, something the original singing DDPS paper's authors did not consider. Their largest single voice dataset had a compressed size of 70Mb, whereas the pre-processed datasets used in this thesis were approximately ten times that at 700Mb, whilst still being based on a single vocal artist, style of music, and vocals only tracks. Using a more extensive training dataset would reduce over-fitting and aid generalisation. The remaining vocal tracks were then pre-processed.

3.1.2 Pre-processing

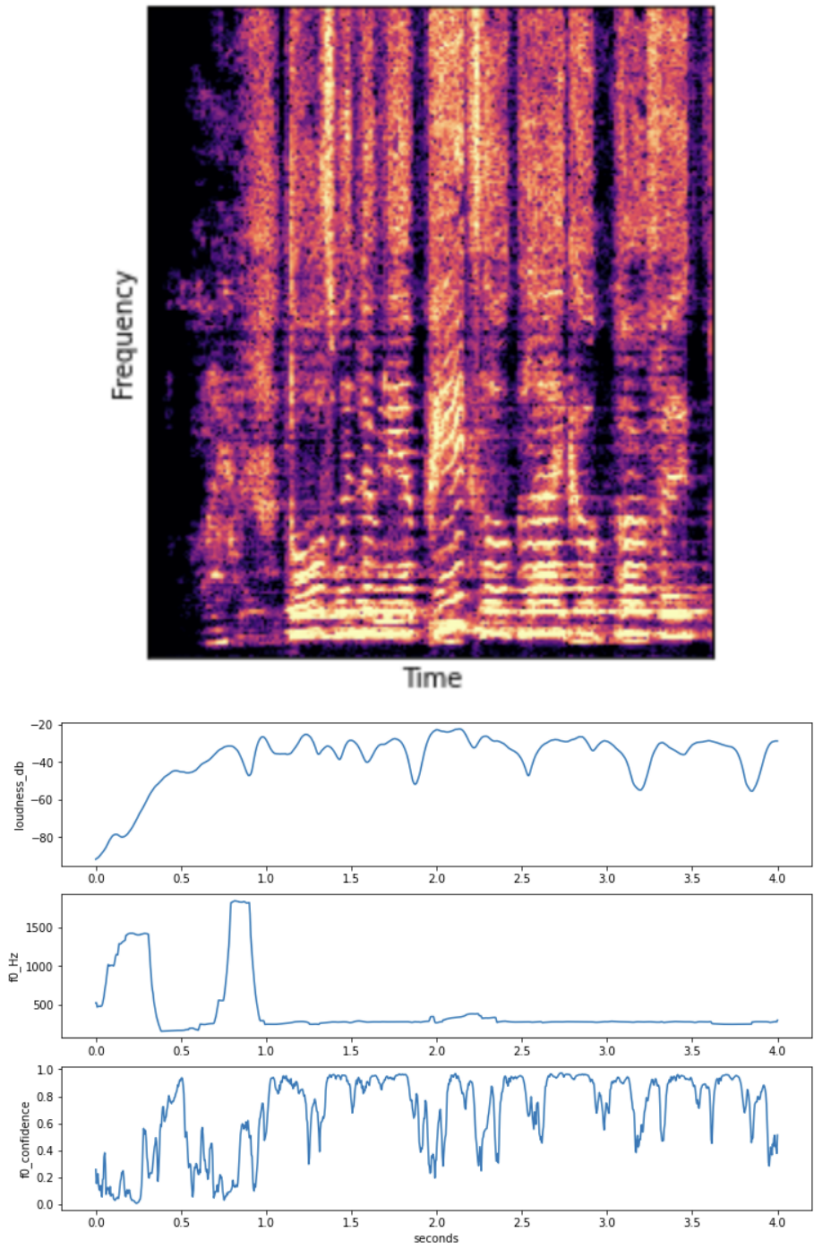

Figure 3.2: Dataset Pre-processing: Spectrogram plot of a random 4 second sample from one of the datasets and its accomanying F0, F0 Confidence and Amplitude characteristics over time throughout the sample

Pre-processing the datasets involved splitting the raw audio into smaller frames (samples), each 4 seconds long. The frame length was limited to 4 seconds to avoid capturing too much information in one spectrogram, which would make learning using the convolutional neural network difficult.

For each frame, F0 and confidence of F0 probability were inferred using CREPE[16]. Amplitude was computed statistically using the Librosa library[18]. Latent Z information was available by passing the raw audio through the encoder. The 4-second samples and accompanying features were then stored as TFRecord files.

Each of the two datasets was pre-processed on Google Colab notebooks; this process took approximately 40 minutes for each dataset using an NVIDIA Tesla V100 GPU.

Finally, a random 4-second clip was selected from each dataset to prove successful pre-processing. Its spectrogram was computed and plotted. Computed F0, F0 Confidence and Amplitude characteristics were also plotted for the selected clip. The underlying audio sample could also be played.

3.2 Training

An additional preprocessor was used to resample the fundamental frequency and loudness, taking into account the sample rate, frame rate, and the number of timesteps hyperparameters. The number of timesteps was set at 1000 per 4-second clip, giving a spectral resolution of 4ms per timestep. This timestep amount was deemed the best compromise between computational efficiency and accuracy.

The standard DDSP encoder-decoder setup was used except with the additional MFCC layer in the decoder as described previously in Singing Voice Synthesis Using DDSP.

The model settings were kept the same as in Singing Voice Synthesis Using DDSP as they had already validated their hyperparameter selection. Their hyperparameter selection included the use of 100 sinusoidal harmonic components and 60 filter banks; this was done to limit model size to ensure it fitted on one GPU.

Each model was trained for 200,000 epochs; the training time was approximately 2.9 epochs per second or 5.2 epochs per second, depending on the GPU. Total training time was approximately 20 hours per model.

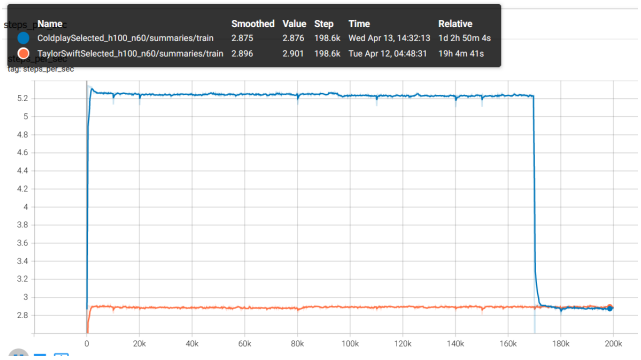

Figure 3.3: Training steps per second, on best hardware available in Google Colab sufficient training to 200,000 epochs could be achieved in approximately 11 hours (assuming 5 epochs per second).

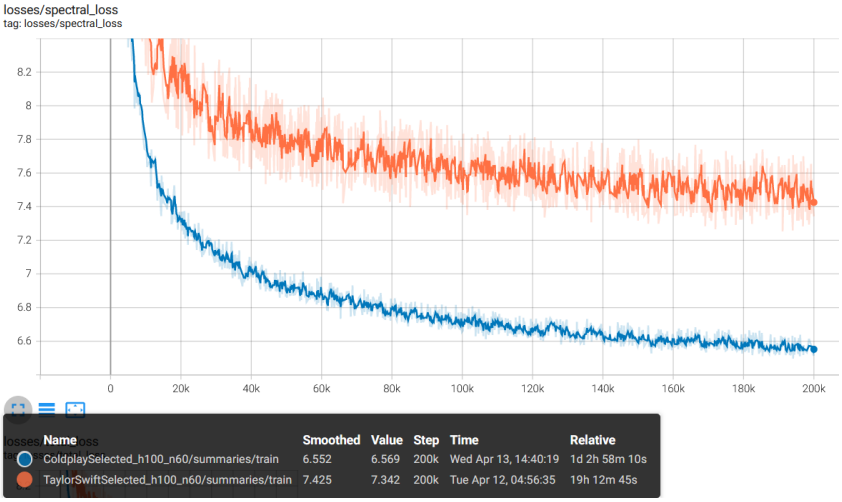

Figure 3.4: Training Spectral Losses: Losses over the 200,000 epochs of training both models, using the spectral loss function defined in Measure of Loss

Figure 3.4 shows the Coldplay losses being less than the Taylor Swift losses. The Coldplay Dataset has more pure silence frames, meaning that the Coldplay model fitted the silence frames more accurately. Furthermore, the Coldplay dataset was smaller than the Taylor Swift one.

3.3 Results

Several inferencing tests were conducted to see if the models had learned the latent features correctly to evaluate the models' performance.

All inferencing tests took approximately 0.3 seconds for a 4-minute frame; this is fast enough to enable real-time applications and significantly quicker than the Jukebox model.

3.3.1 Recreation from the Training Dataset

A random frame was selected and passed through each model; loudness and F0 were unmodified. The results of the inferencing were then compared to the original training dataset.

Each model successfully recreated original frames, though the Taylor Swift dataset yielded the best results.

Timbral features were slightly distorted (more so with the Coldplay model), but the overall quality was good, and it was easy to tell it was the original singer in both cases. In addition, pitch estimation was highly accurate and in line with the original pitch. Pitch accuracy can be partially attributed to the accuracy of the CREPE pitch detection model[16] and because the pitch was directly passed to the harmonic synthesiser.

The resynthesis of understandable words from the original frame was a far more considerable achievement. The original singing DDSP paper[1] suffered a problem of stuttering when attempting resynthesis as their model was unable to recreate phonemes of the human voice accurately. It is possible that using far larger datasets has improved the quality of the resynthesis because the models had become more general in their ability to synthesise the human voice. However, it must be said that the Coldplay model was harder to understand.

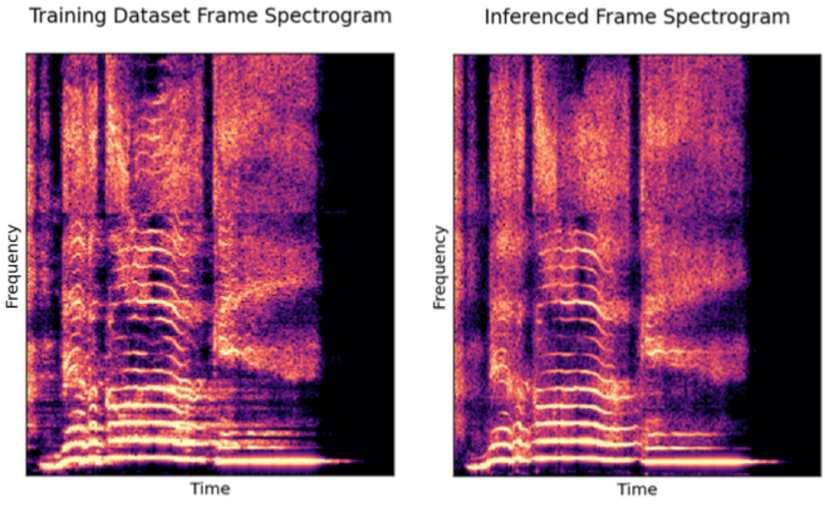

Figure 3.5: (Taylor Swift) Original and resynthesized frames without latent modification

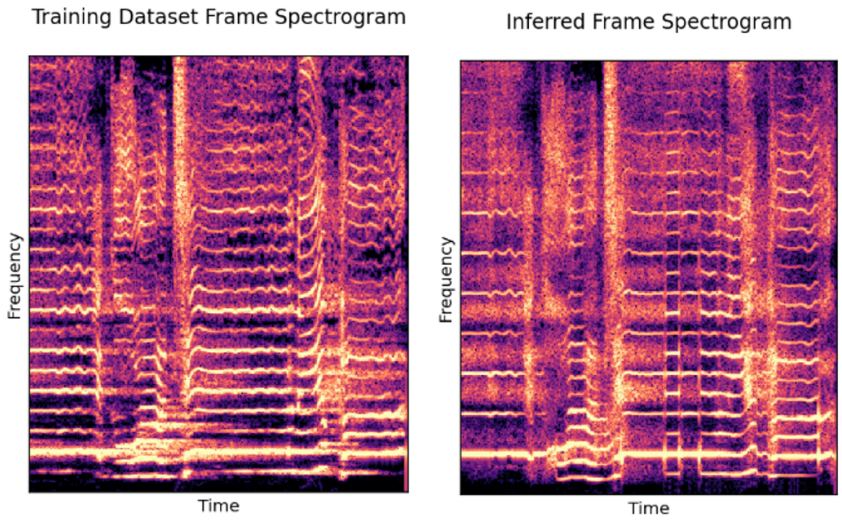

Figure 3.6: (Coldplay) Original and resynthesized frames without latent modification

3.3.2 F0 Pitch Transposition by a fixed octave

A more advanced inferencing test was then undertaken. The fundamental frequency latent as determined by CREPE was transposed by fixed octaves (-2, -1, 0, 0.5, 1, 2), and the inferencing was re-performed on the transposed latents. Both models responded to the change in F0 and could accurately transpose harmonic pitch by the correct octave amount. At minor transpositions, e.g. +1 or -1 octave, the inferred frame still sounded somewhat like a human voice. However, it sounded like the noise, and harmonic components were separate sources at greater transpositions and did not constitute one voice.

As expected, modifying F0 did not change the pitch of the filtered noise at all, confirming that the pitch change had been directly passed to the harmonic synthesiser. This finding is interesting because we would have expected a little bit of distortion in the original pitch.

At extreme transpositions, harmonics sometimes appeared to go silent. The resynthesis sounded like a whisper, coming almost entirely out of the filtered noise. Whispering occurs when the vocal cords are held rigid, preventing them from vibrating and producing sinusoidal sounds (harmonics in the case of singing). The fact that the model's output sounded like a whisper is a good sign that the model was able to learn and distinguish the noise and harmonics as per the Harmonic Plus Noise Model[24][7].

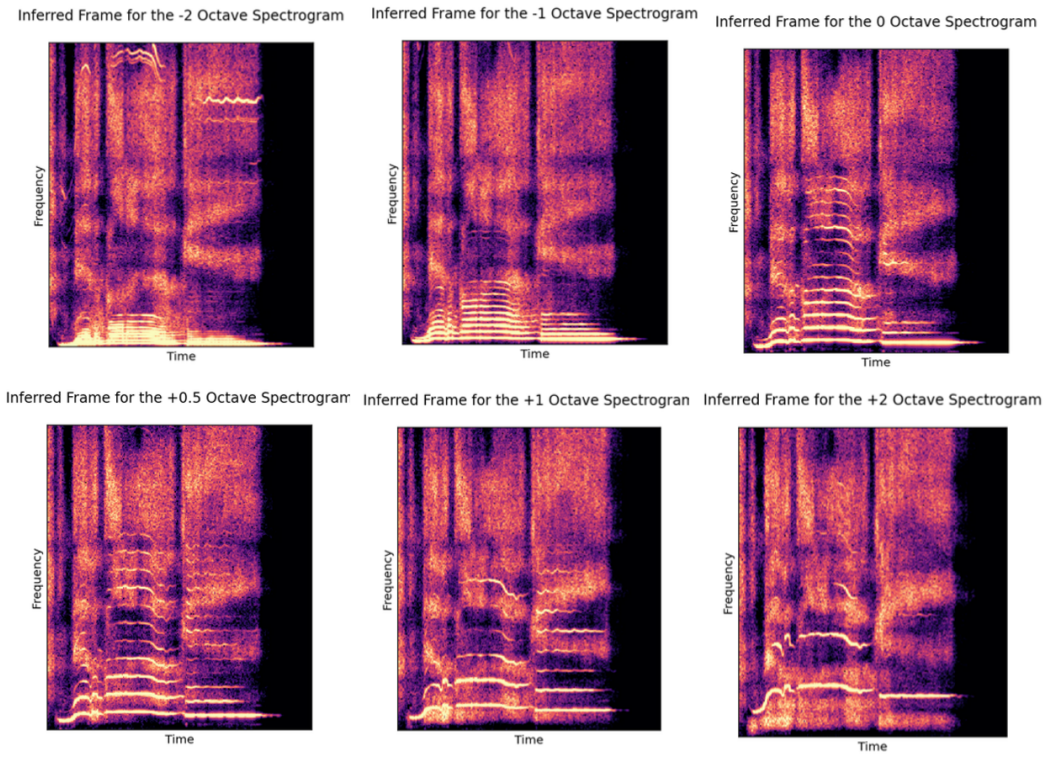

Figure 3.7: (Taylor Swift) Inferred spectrogram frames at various octave transpositions realative to F0 at a certain timegrame in the original frame

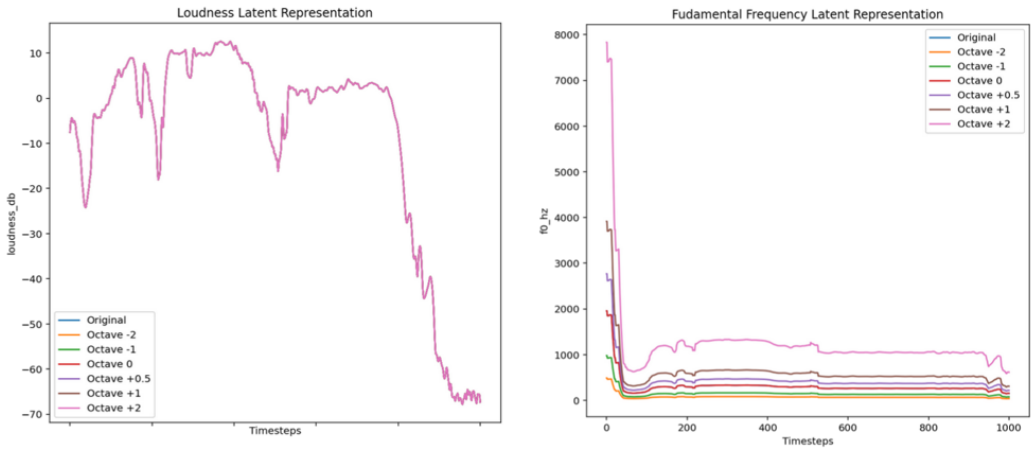

Figure 3.8: (Taylor Swift) Latent F0 and loudness features for various octave transpositions realative to F0 over timesteps throughout the frame

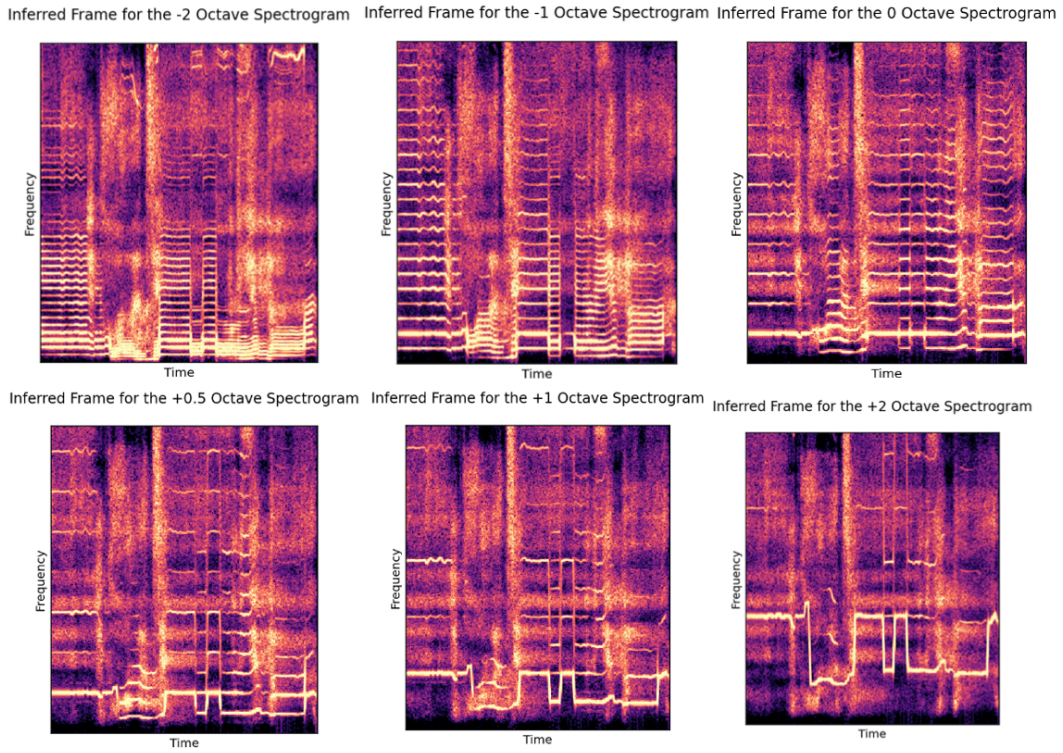

Figure 3.9: (Coldplay) Inferred spectrogram frames at various octave transpositions realative to F0 at a certain timegrame in the original frame

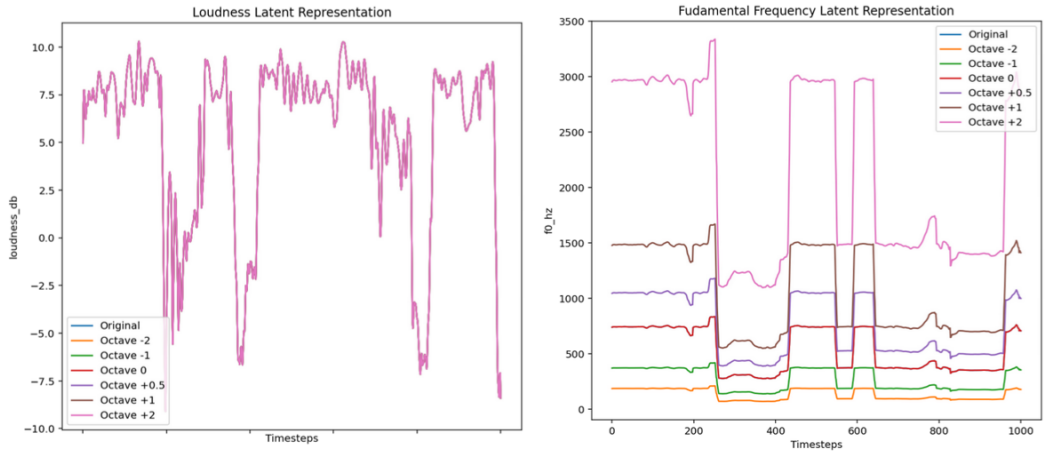

Figure 3.10: (Coldplay) Latent F0 and loudness features for various octave transpositions realative to F0 over timesteps throughout the frame

3.3.3 Fixing F0

The final pitch related test was fixing F0 to the mean value of F0 throughout the frame to see how the models would perform under unnatural pitch conditions.

Both models could fix the pitch to the mean value of F0 in the frame. This fixing is heard and can be seen visibly from the inferred spectrogram images where the harmonic components are at lines of constant frequency, unlike the original where they vary.

Additionally, words were still able to be synthesised and accurately heard, suggesting the model was able to learn the underlying phonemes of speech.

This result is excellent, mimicking what was founded in the speech DDSP research[8] (whose code was not publicly available). Further experiments were done with similar degrees of success in varying the pitch to other amounts, e.g. 100Hz and 500Hz. Again, however, the quality of results broke down at the extremes. The breakdown was evidenced by the harmonic again sounding more like a separate sound, with the whispers producing the actual words.

Sadly, timbral quality was reduced when F0 was fixed, with the output sounding more robotic and the original timbre being lost. Timbral loss is to be expected, however, as the original harmonic plus noise model was not designed with timbre transfer specifically in mind[7].

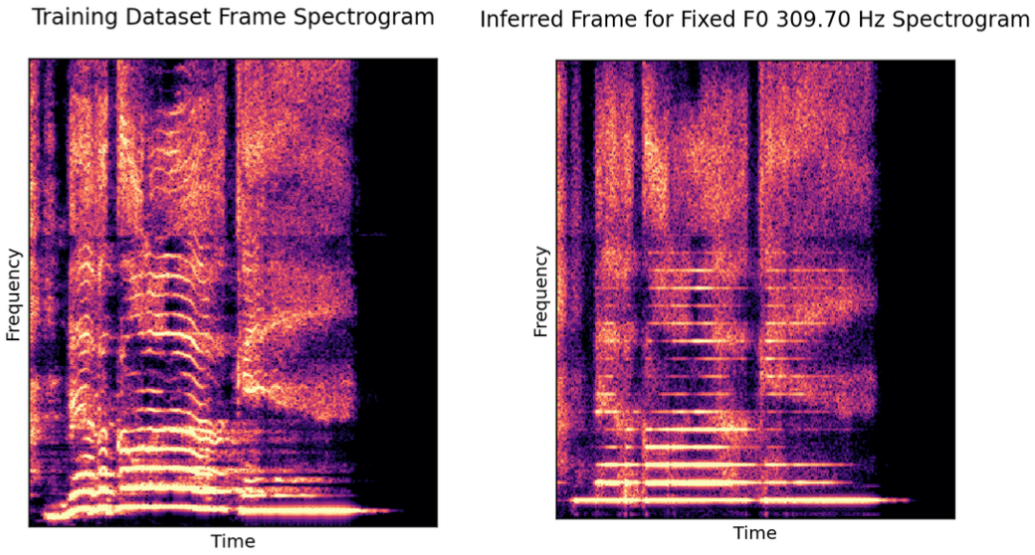

Figure 3.11: (Taylor Swift) Training dataset and fixed F0 spectrogram frames

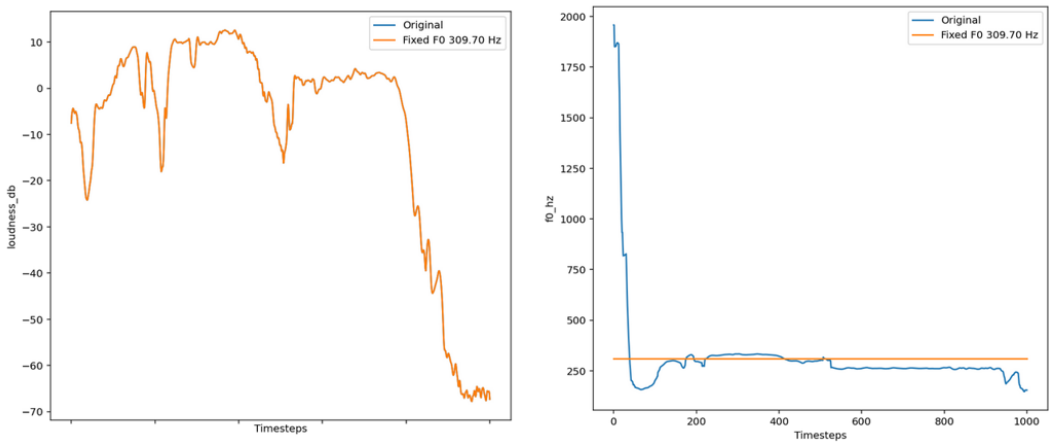

Figure 3.12: (Taylor Swift) Latent information on loudness and F0 over timesteps throughout the frame. The mean F0 was used to fix F0 throughout the frame

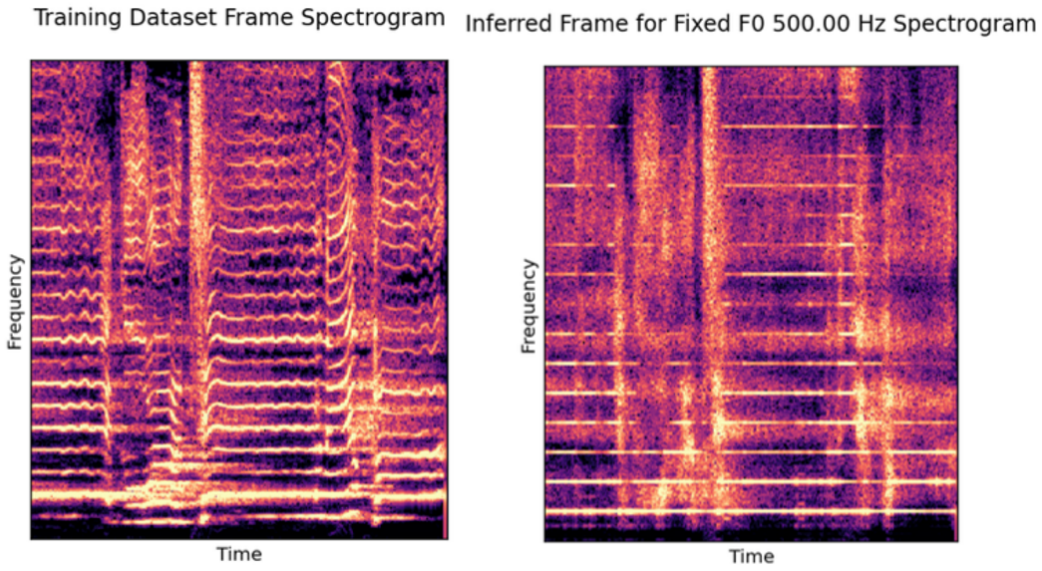

Figure 3.13: (Coldplay) Training dataset and fixed F0 spectrogram frames

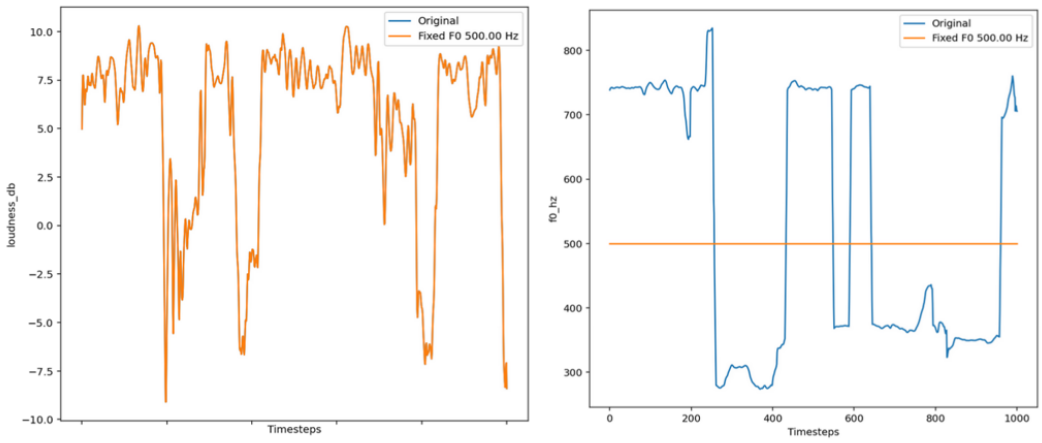

Figure 3.14: (Coldplay) Latent information on loudness and F0 over timesteps throughout the frame. The mean F0 was used to fix F0 throughout the frame

3.3.4 Modifying Loudness

Unfortunately, modification of the loudness latent vector did not alter the loudness of any inferred frames, and this could be caused by, perhaps, the decoder learning to ignore the latent loudness vector.

Alternatively, there may be some architectural issues with the modified decoder. For example, the DDSP library's authors significantly redesigned the loudness and power calculations with the version 3 release. Unfortunately, the Singing DDSP decoder used the older version 1. The discovery of any technical problems is beyond the scope of this thesis.

3.3.5 Timbral Transfer

The Taylor Swift model was selected for the timbral transfer tasks due to its better performance in the pitch transfer tests. For both male and female voice tests, vocals were successfully synthesised, and it was clear what words were being pronounced. Pitch was also accurately modelled for both cases. Success outside of the training set suggests that the model had successfully been generalised and had learned the underlying phonemes of human speech from spectrograms.

For the timbral transfer, the following songs were used:

- **(Male Voice) Lewis Capaldi** - ***Someone You Loved***
- **(Female Voice) Birdy** - ***Deep End***

Sadly, the timbral transfer did not occur, with each inferred frame sounding similar to the source artist and not that of the training dataset. This result suggests that the model had learned to transfer timbre from the original vocal frame to the synthesised frame. Timbral quality was also reduced, sounding more unnatural, suggesting that perfect generalisation was not fully achieved.

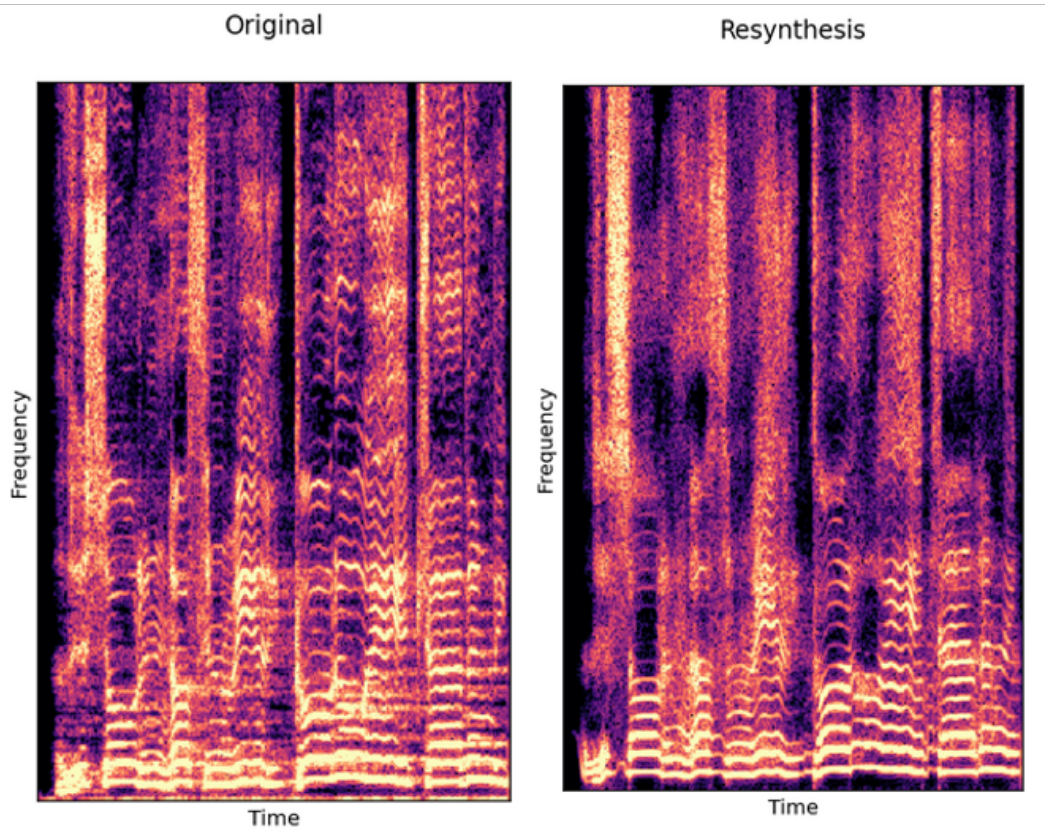

Figure 3.15: Lewis Capaldi timbral transfer test showing a comparison between the orignal and infererd spectrogram frames using the Taylor Swift model

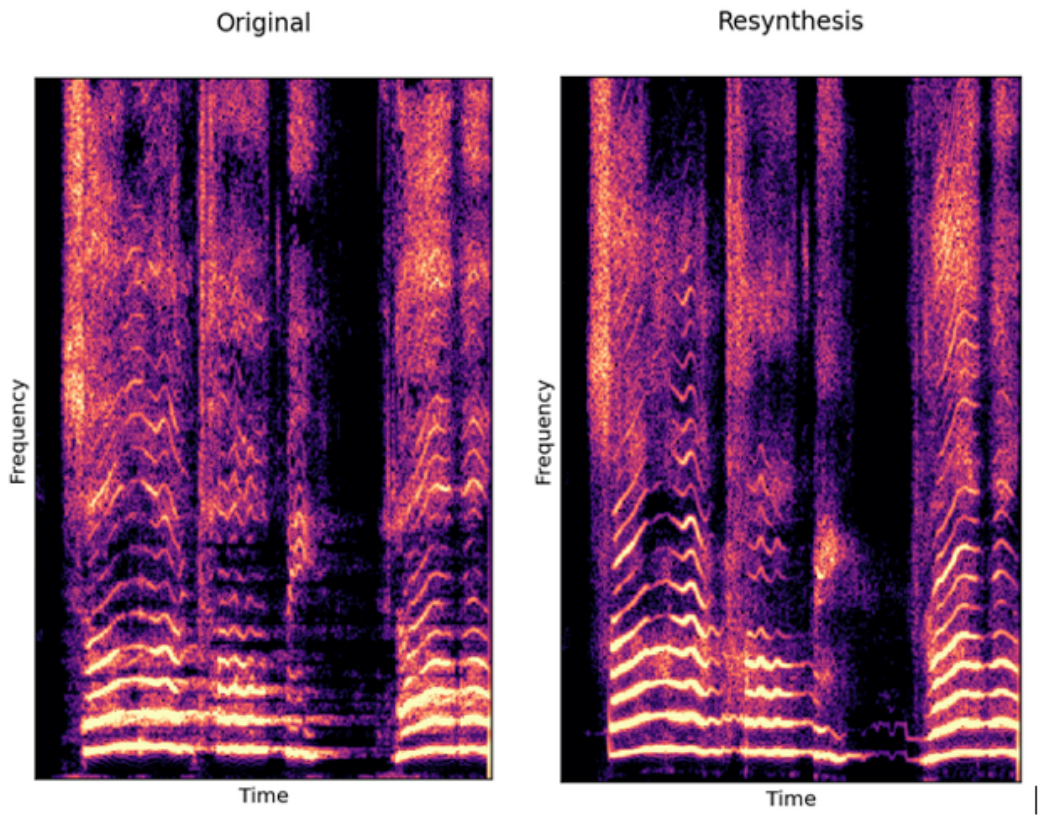

Figure 3.16: Birdy timbral transfer test showing a comparison between the oriignal and infererd spectrogram frames using the Taylor Swift model

3.3.6 Inference with Instrumentals

The Taylor Swift model was then used to infer a frame featuring the instrumental and vocal version of the song Deep End (i.e. the original song before passing through the Spleeter model).

The DDSP model managed to extract vocals from the track with instrumentals. Even more significant, there was no leakage of instrumentals in the inferred frame, suggesting that the model had isolated features unique to the human voice instead of more general noise features.

This result shows that the DDSP model is very flexible, being able to carry out source separation tasks similar to the Spleeter library[12]; this was not demonstrated in the previous DDSP papers.

The inferred frames from the tracks with and without instrumentals sounded similar; however, F0 estimation in the frame with instrumentals could be better. Inaccurate F0 estimation is likely a limitation of using CREPE pitch estimation on a track with multiple parts simultaneously.

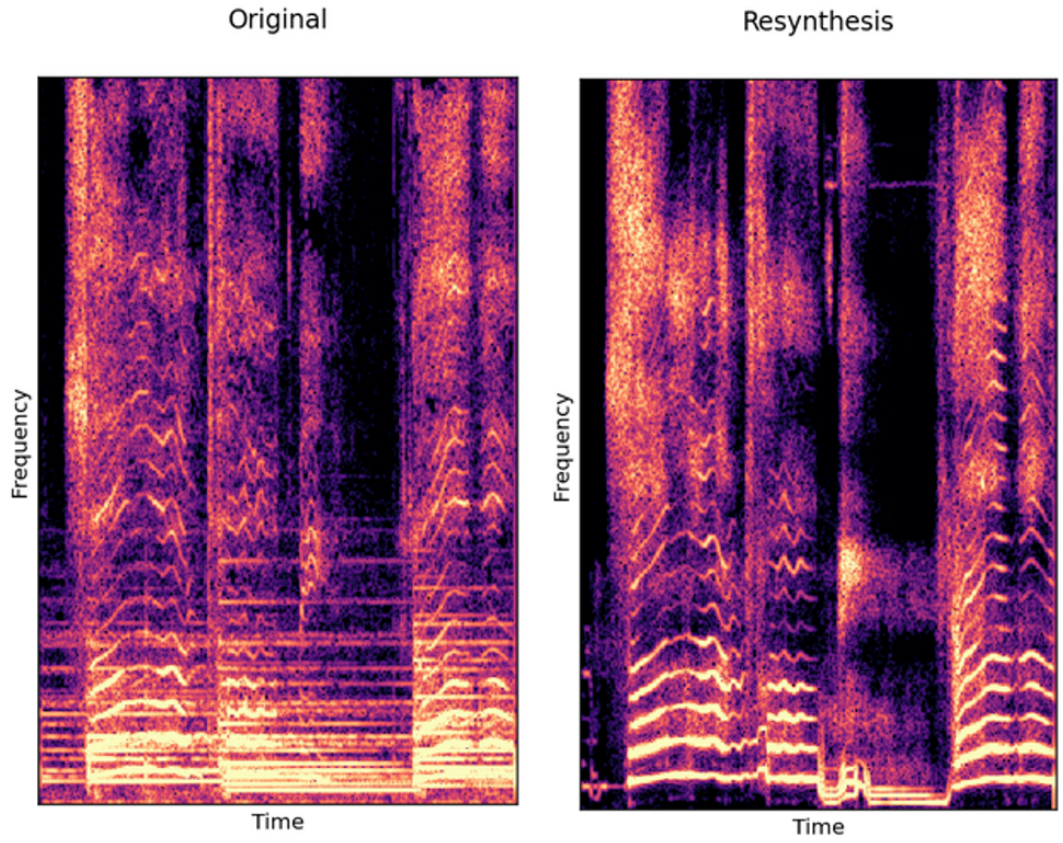

Figure 3.17: Birdy instrumental and vocals inference test using the Taylor Swift model, showing the original and infered spectrogram frames

3.3.7 General Problems

Along with the loudness perception problem, others were encountered during all inferencing tests:

1. At low loudness levels, the CREPE model could not accurately detect the pitch of the audio sample. This lack of accuracy caused the fundamental frequency to jump around, sometimes appearing to jump up or down an octave, making the output sound jarring; this is evidenced in the Coldplay frames that all experience this problem. The DDSP authors suggested a potential solution was muting sections of the track where CREPE had low confidence in its prediction. Sadly, this fix could not work here due to the problems discussed around the loudness latent.
2. The models failed to learn specific timbre related to the trained artist; this was especially apparent during the constant F0 test when the output sounded almost robotic. Lack of timbre transfer was foreseen as the DDSP architecture was not directly designed for timbre transfer.

3. Similarly, when transferring timbre from another artist, the inferred sample sounded like the voice of the other artist more than it did that of the training dataset artist. However, this result could be attributed to the model's generalisation to synthesise the specific timbres derived from the noise characteristics of a test frame.

The failure to learn timbre, as demonstrated throughout all the tests, could be the price paid for the ability of the model to understand, interpret, and synthesise any given voice.

4 Conclusions and Reccomendations

4.1 Experimental Conclusions

In summary, DDSP has demonstrated itself as a powerful tool for learning and synthesising the vocal features of the human voice. The experiments shown in this thesis demonstrated how it could accurately infer pitch and phonetic information about the human voice. This result is significant as it appears to overcome some of the problems of the original MFCC DDSP model[1] which was unable to learn the underlying phonemes of speech and produce coherent speech. Furthermore, generality outside of the training dataset was demonstrated in many situations.

It could be said that despite the failure of the models in this thesis to transfer timbre and loudness, the interpretable pitch feature of DDSP has been successfully learned and demonstrated.

Further applicability previously not explored in the DDSP paper, such as vocal source separation, has also been demonstrated. The model had successfully learned to infer only the vocal source from the audio signal.

DDSP is already one of the best deep learning architectures for synthesising realistic music vocals. It will likely be used for future academic or artistic purposes. Its interpretable and modular nature will allow it to be integrated into other applications. Further research into the alternatives, some of which were presented in this thesis, e.g. Jukebox is, however, required to evaluate if DDSP is the best.

4.2 Reccomendations

However, there are many areas of improvement that can be made to the model. Firstly, accurate loudness estimation must be implemented. Loudness control is a feature of the original DDSP model[7] that appears not to be working in the model used in this paper. Therefore, a solution may require an extensive decoder review. If it is found to be working as desired, the loss function may have to be altered to bias the model towards the amplitude latent vector.

The timbral quality, although good, could be better. Timbral quality could be improved by increasing the number of internal parameters and increasing the resolution of spectrograms used. However, this change would increase the model size significantly, requiring its parallelisation and training on multiple GPUs. Furthermore, timbral transfer like that demonstrated in the original DDSP paper did not work for the models trialled in this thesis. Future work could pertain to addressing this discrepancy.

Future works could implement DDSP with other technologies and machine learning models. For example, a DDSP based model could be combined with an attention-based symbolic transformer model, a natural language model such as GPT-3[2] and text to speech systems to provide an end to end program that could generate a whole song. The symbolic transformer model could generate long-term features such as F0 over time (in the form of midi notation) and other long-term musical features and characteristics. The natural language model could then be tasked with generating lyrics to match the timings of the generated music. These could be synthesised using text to speech and F0 or other characteristics modified by DDSP.

As suggested by previous work[1], native phonetic conditioning could be implemented into DDSP, enabling modification of the underlying words and phonetics as latents similarly to F0 or loudness. Implementing this would enable further expressibility in the DDSP architecture and avoid the need for any other text to speech model.

Acronyms

ANN	Artificial Neural Network
CNN	Convolutional Neural Network
CREPE	Convolutional Representation for Pitch Estimation
DDSP	Differentiable Digital Signal Processing
DFT	Discrete Fourier Transform
DNN	Deep Neural Network
FFT	Fast Fourier Transform
IDFT	Inverse Discrete Fourier Transform
L1	Least Absolute Deviations
LTI-FIR	Linear Time Invariant Finite Impulse Response Filter
MFC	Mel Frequency Cepstrum
MFCC	Mel Frequency Cepstral Coefficients
RNN	Recurrent Neural Network
SMS	Spectral Modelling Synthesis
STFT	Short-Time Fourier Transform
VST	Virtual Studio Technology

Open Source Licenses

1. Latex Mimososis Latex Template[21]
2. DDSP (Differentiable Digital Signal Processing) Python Library[3]
3. Spleeter - Source Separation Library[23]

Bibliography

1. J. Alonso and C. Erkut. *Latent Space Explorations of Singing Voice Synthesis using DDSP*. https://arxiv.org/pdf/2103.07197.pdf. 2021.
2. T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. "Language Models are Few-Shot Learners". *CoRR* abs/2005.14165, 2020. arXiv: 2005.14165. URL: https://arxiv.org/abs/2005.14165.
3. *ddsp*. [Version 3.3.2; accessed 31-March-2022]. URL: https://pypi.org/project/ddsp/3.3.2/.
4. P. Dhariwal, H. Jun, C. Payne, A. R. Jong Wook Kim, and I. Sutskever. *Jukebox*. https://openai.com/blog/jukebox/. 2020.
5. P. Dhariwal, H. Jun, C. Payne, A. R. Jong Wook Kim, and I. Sutskever. *Jukebox: A Generative Model for Music*. https://arxiv.org/abs/2005.00341. 2020.
6. J. engel, K. K. Agrawal, S. Chen, I. Gulrajani, C. Donahue, and A. Roberts. *GANSynth: Generative Adversarial Neural Audio Synthesis*. https://openreview.net/pdf?id=H1xQVn09FX. 2019.
7. J. Engel, L. Hantrakul, C. Gu, and A. Roberts. *DDSP: Differentiable Digital Signal Processing*. https://arxiv.org/abs/2001.04643. 2020.
8. G. Fabbro, V. Golkkov, T. Kemp, and D. Cremers. *SPEECH SYNTHESIS AND CONTROL USING DIFFERENTIABLE DSP*. https://arxiv.org/pdf/2010.15084.pdf. 2020.
9. T. Ganchev, N. Fakotakis, and G. Kokkinakis. "Comparative evaluation of various MFCC implementations on the speaker verification task". *10th International Conference on Speech and Computer (SPECOM 2005)* Vol. 1, 2005.
10. *Getting to Know the Mel Spectrogram*. https://towardsdatascience.com/getting-to-know-the-mel-spectrogram-31bca3e2d9d0. [Online; accessed 24-March-2022]. 2019.
11. U. Grenandder. *Probability and Statistics: The Harald Cramer Volume*. Wilehy, 1959.
12. R. Hennequin, A. Khlif, F. Voituret, and M. Moussallam. "Spleeter: a fast and efficient music source separation tool with pre-trained models". *Journal of Open Source Software* 5:50, 2020, p. 2154. DOI: 10.21105/joss.02154. URL: https://doi.org/10.21105/joss.02154.

13. L. A. Hiller Jr. and L. M. Isaacson. "Musical Composition with a High-Speed Digital Computer". *J. Audio Eng. Soc* 6:3, 1958, pp. 154–160. URL: http://www.aes.org/e-lib/browse.cfm?elib=231.

14. C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck. *Music Transformer: Generating Music with Long-Term Structure*. https://arxiv.org/pdf/1809.04281.pdf. 2018.

15. L. J. *Creation by refinement: a creativity paradigm for gradient descent learning networks*. https://ieeexplore.ieee.org/author/37087959474. 1988.

16. J. W. Kim, J. Salamon, P. Li, and J. P. Bello. *CREPE: A Convolutional Representation for Pitch Estimation*. 2018. DOI: 10.48550/ARXIV.1802.06182. URL: https://arxiv.org/abs/1802.06182.

17. Landr. *What Is MIDI? How To Use the Most Powerful Tool in Music*. 2022. URL: https://blog.landr.com/what-is-midi/.

18. *librosa*. [Version 0.9.1; accessed 26-February-2022]. URL: https://pypi.org/project/librosa/0.9.1/.

19. M. Marolt, A. Kavcic, and M. Privosnik. *Neural Networks for Note Onset Detection in Piano Music*. https://www.researchgate.net/publication/2473938_Neural_Networks_for_Note_Onset_Detection_in_Piano_Music. 2003.

20. D. O'Shaughnessy. *Speech communication: human and machine*. Addison-Wesley Publishing Company, 1987.

21. Pseudomanifold. *latex-mimosis*. https://github.com/Pseudomanifold/latex-mimosis. [Online; accessed 23-March-2022; commit 339228e].

22. X. Serra and J. Smith. "Spectral Modeling Synthesis: A Sound Analysis/Synthesis System Based on a Deterministic Plus Stochastic Decomposition". *Computer Music Journal* 14:4, 1990, pp. 12–24. ISSN: 01489267, 15315169. URL: http://www.jstor.org/stable/3680788.

23. *spleeter*. [Version 2.3.0; accessed 02-January-2022]. URL: https://pypi.org/project/spleeter/2.3.0/.

24. Y. Stylianou. "Modeling Speech Based on Harmonic Plus Noise Models". In: *Nonlinear Speech Modeling and Applications*. Ed. by G. Chollet, A. Esposito, M. Faundez-Zanuy, and M. Marinaro. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 244–260. ISBN: 978-3-540-31886-6.

25. P. M. Todd. *A Connectionist Approach To Algorithmic Composition*. https://www.jstor.org/stable/3679551?seq=1. 1989.

26. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. *Attention Is All You Need*. https://arxiv.org/abs/1706.03762. 2017.

27. Wikipedia contributors. *Taxicab geometry — Wikipedia, The Free Encyclopedia*. [Online; accessed 23-March-2022]. 2022. URL: https://en.wikipedia.org/w/index.php?title=Taxicab_geometry&oldid=1066973902.