Imperial College London

Department of Mathematics

Cascaded Diffusion Models for Speech Synthesis

Haaris HAYAT

CID: 00640637

Supervised by Kevin WEBSTER

4 September 2023

Submitted in partial fulfilment of the requirements for the
MSc in Machine Learning and Data Science of
Imperial College London

The work contained in this thesis is my own work unless otherwise stated.

Signed: Haaris Hayat Date: 4 September 2023

Abstract

Speech is a fundamental aspect of communication and text to speech models play an ever increasing role in our lives. In an attempt to enable us to better interact with technology, products such as mobile phones, Alexa, Google Home, cars etc. now feature text to speech capabilities. However text to speech can play an even more fundamental role in people's lives such as helping people with paralysis to communicate through speech. In the last five to six years, there has been a big push in research to improve text to speech models using deep learning techniques, and to not just improve their accuracy but to make them sound more "human".

In this thesis, we propose a novel approach to generate speech using denoising diffusion probabilistic models that learn to transform white noise to speech data. We build upon the current research in speech generation using denoising diffusion probabilistic models, and combine this with the latest research in diffusion models in the image domain. While the need for further research and refinement in this area is evident, our preliminary results show the viability of our approach and highlight how our approach could address some of the issues pertaining to speech generation models.

Acknowledgements

I would like to thank my supervisor, Kevin Webster, for providing his support, time, advice and for inspiring an interest in Deep Learning and the audio domain. I would also like to thank my tutor, James Martin, for all the academic and moral support.

Finally, I would like to thank Anousha, Yahya and Zakariya for their patience during the last two years.

Contents

1 Introduction **1**

2 Basics of Audio Data **3**
2.1 Key Terminology . . . 3
2.2 Mel Spectrograms . . . 5
2.2.1 Fourier Transformations . . . 5
2.2.2 Spectrogram Generation . . . 6
2.2.3 Mel-scale Conversion . . . 6
2.2.4 Advantages and Disadvantages . . . 7

3 Background to Diffusion Models **8**
3.1 Overview . . . 8
3.2 Diffusion Model Details . . . 9
3.2.1 Forward Diffusion Process . . . 9
3.2.2 Reverse Diffusion Process . . . 10
3.2.3 Training . . . 10
3.2.4 Sampling . . . 13
3.2.5 Conditioning . . . 13
3.3 Network Architecture: U-Net . . . 13
3.3.1 U-Nets in diffusion models . . . 14
3.4 Cascaded Diffusion Models . . . 15

4 Recent Developments in the Field of Speech Synthesis **18**
4.1 Autoregressive Models . . . 18
4.1.1 WaveNet (Oord et al., 2016) . . . 18
4.1.2 Tacotron . . . 20
4.1.3 Transformer-TTS (Li et al., 2019) . . . 21
4.2 Generative Adverserial Networks . . . 21
4.2.1 WaveGAN and SpecGAN (Donahue et al., 2018) . . . 22
4.2.2 HiFi-GAN (Kong et al., 2020a) . . . 23
4.3 Diffusion Models . . . 24
4.3.1 DiffWave (Kong et al., 2020b) . . . 24
4.3.2 Grad-TTS (Popov et al., 2021) . . . 25
4.4 Potential Impact of using Cascaded Diffusion Models . . . 26

5 Methodology **28**
5.1 Text Encoder . . . 28
5.2 Duration Predictor . . . 29
5.2.1 Monotonic Alignment Search . . . 30
5.2.2 Montreal Forced Alignment . . . 31
5.2.3 Training the Duration Predictor . . . 32
5.2.4 Applying Durations to Embeddings . . . 32
5.3 Bi-directional Dilated Convolution Based Model . . . 33
5.3.1 Diffusion step embedding . . . 34
5.3.2 Conditioner . . . 35
5.3.3 Residual Layer Architecture . . . 35
5.3.4 Final Projection . . . 36
5.3.5 Choosing the Amount of Dilation . . . 36
5.3.6 Fast Sampling . . . 37
5.4 Super Resolution Models . . . 37
5.5 Dataset . . . 38
5.6 Evaluation Metrics . . . 38
5.6.1 Mean Opinion Score . . . 38
5.6.2 Fréchet Audio Distance . . . 39
5.6.3 Other Metrics . . . 40
5.7 Training . . . 41

6 Experiments **43**
6.1 Experiment I: Cascaded Diffusion Models Involving Low Resolution Mel Spectrograms . . . 43
6.1.1 Data Pre-processing . . . 44
6.1.2 Approach . . . 44
6.1.3 Results . . . 45
6.2 Experiment II: Text to Speech (End to End) Using Cascaded Diffusion Models . . . 46
6.2.1 Data Pre-processing . . . 47
6.2.2 Approach . . . 47
6.2.3 Results . . . 49
6.3 Experiment III: Direct Waveform Generation Using Cascaded Diffusion Models . . . 51
6.3.1 Base Model . . . 52
6.3.2 Super Resolution . . . 54
6.3.3 Results . . . 54

7 Conclusion and Further Research **55**
7.1 Conclusion . . . 55
7.2 Further Research . . . 55
7.2.1 Conditional Augmentation . . . 55
7.2.2 Mel Spectrogram Super Resolution . . . 56

7.2.3 Experimentation with Larger Models 56
7.2.4 Generalisation of Approach 56

1 Introduction

The landscape of speech generating models has evolved drastically in the last five years with the introduction of deep learning techniques. There are various types of speech generating models such as: text to speech models (sometimes referred to as end to end speech generating models) which take text as input and use this to generate speech; and vocoders which take spectrograms as inputs to generate speech. Spectrograms are lower detail representations of waveforms which show the frequencies of an audio signal and how these frequencies change over time (See Section 2.2 for further details). Note that most text to speech models will use a two step approach by generating spectrograms first before using these to generate the speech.

The earliest models were autoregressive models which proved highly accurate but suffered from extremely long inference times. Subsequent attempts tried to reduce these inference times whilst maintaining the quality levels.

In the last two years, denoising diffusion probabilistic models (referred to as diffusion models for brevity), which learn to transform noise to meaningful data have led the research in speech generation. This new family of models not only exceeds previous quality standards but is able to generate speech in acceptably low inference times. The most recent development in diffusion models has been to use a pipeline, called a cascade of diffusion models (Ho et al., 2022) which produce data of increasing resolutions and outperform regular diffusion models when it comes to image generation.

To the best of our knowledge, cascaded diffusion models have not been trialled to generate speech and the overall goal of this thesis is to investigate the use of such models in speech generation. We will consider the use of cascaded diffusion models in both the context of text to speech models and vocoders.

In order to delve deeper into our research goal, it is important to understand some of the ways that current speech generating models can be improved. This will help guide our research, the experiments we conduct, as well as the evaluation techniques used to assess the viability of our proposed approach. We believe that there are two key ways that diffusion based speech generating models can be improved, and these form our sub-goals:

1. Improving the quality of the generated speech. This is probably the most important factor in speech generation; how well a model can generate speech that resembles speech spoken by a human, has correct pronunciation, and utilises variation in tones/pace according to the context.

2. Making it easier to train diffusion based speech generating models so that they can be adapted to new datasets more easily. Currently, one of the major issues with diffusion models is the large amount of time and computational power required to train these large models.

In addition, we will experiment with generating waveforms directly from text (as opposed to generating spectrograms first). This is a difficult task, and one which researchers have struggled to solve with the current deep learning techniques (See (Donahue et al., 2018) and (Popov et al., 2021)).

The rest of this thesis is structured as follows: in Chapter 2, we provide some preliminary information relating to the audio domain, and in Chapter 3 we explain the details behind diffusion models; how they work, how they are trained; and how they generate data. In Chapter 4 we discuss the recent developments in the field of speech generation, followed by Chapter 5 where we describe our methodology. The methodology chapter describes the building blocks of our experiments such as the techniques we use, as well as the datasets and evaluation metrics.

In Chapter 6, we will describe how our experiments are designed to address the goals mentioned above before detailing how the building blocks from Chapter 5 are used to conduct these experiments.

Finally, in Chapter 7 we conclude our research and describe any further research that could be conducted (other goals that could be addressed using cascaded diffusion models).

2 Basics of Audio Data

Since audio data can be very different to other types of data, in this Chapter we will spend some time discussing some of the attributes of audio data, key terminology and common pre-processing techniques.

2.1 Key Terminology

Definition 2.1.1. The **waveform** of a sound signal is the shape and form of the signal. More precisely, a single waveform is a 1-dimensional array with amplitude values across time. An example waveform is shown in Figure 2.1. Waveform data is stored in .wav files.

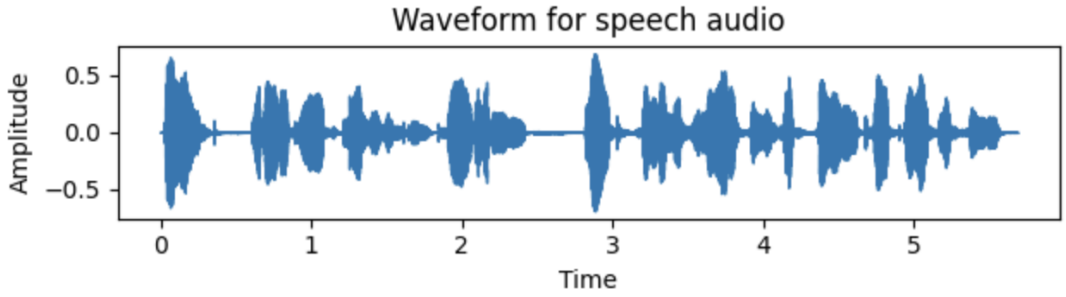

Figure 2.1: Example waveform for speech audio

Definition 2.1.2. The **Sample rate** of audio (in units of Hz) is the number of samples per second. Since audio represents an analog signal, recording this digitally involves taking samples at regular intervals so that the signal can be reconstructed. The higher the sample rate, the closer the signal is approximated and thus the higher the quality of the reconstructed audio.

Figure 2.2 depicts how difference in sample rates can impact the quality of the audio.

The sample rate can be thought of as the equivalent to the resolution of an image. However unlike images, where the total number of pixels are fixed, in audio files only the number of samples per second is fixed and the total number naturally depends on the length of the audio.

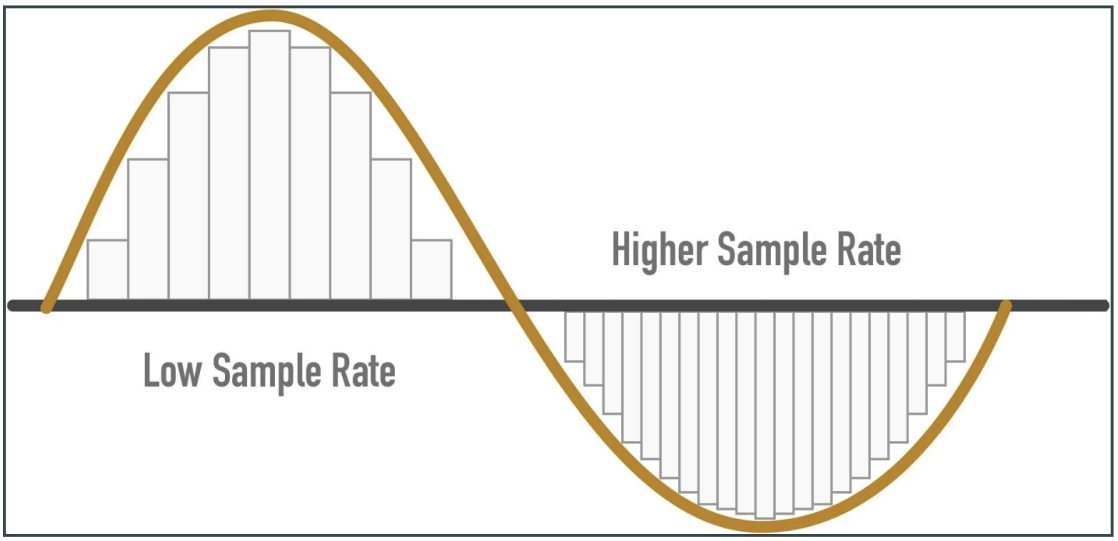

Figure 2.2: Difference between a low sample rate and high sample rate. *Source: www.headphonesty.com*

Definition 2.1.3. The **Bit Depth** is the number of bits of information that is stored for each sample. This is another attribute of audio data that determines the quality of the audio. For example, a bit rate of 16 means the amplitude of a waveform has 2^{16} different levels. For our purpose, when we talk about the resolution we will mean the sample rate and will not be modifying the bit rate.

Definition 2.1.4. Clipping is the impact of having a bit depth that is too low whereby extreme values are capped resulting in distortion of the signal.

Definition 2.1.5. The **Mel scale** is a perceptual scale of pitches deemed to be equal in distance from one another to the human ear. The scale is derived due to the way humans can better ascertain changes in pitches at the lower end of the scale as compared to the higher ends. The conversion from standard frequencies to mel frequencies is shown in Figure 2.3.

Definition 2.1.6. The **Phoneme** is the smallest unit of speech distinguishing one word (or word element) from another.

Definition 2.1.7. Prosody is the pattern of stress, intonation, pitch and tone in spoken language that distinguishes a person's manner of speech.

Definition 2.1.8. The **Mean Opinion Score** or "MOS" is the average score given to audio samples by human listeners on a scale of 1 to 5 in order to assess the quality (5 being excellent).

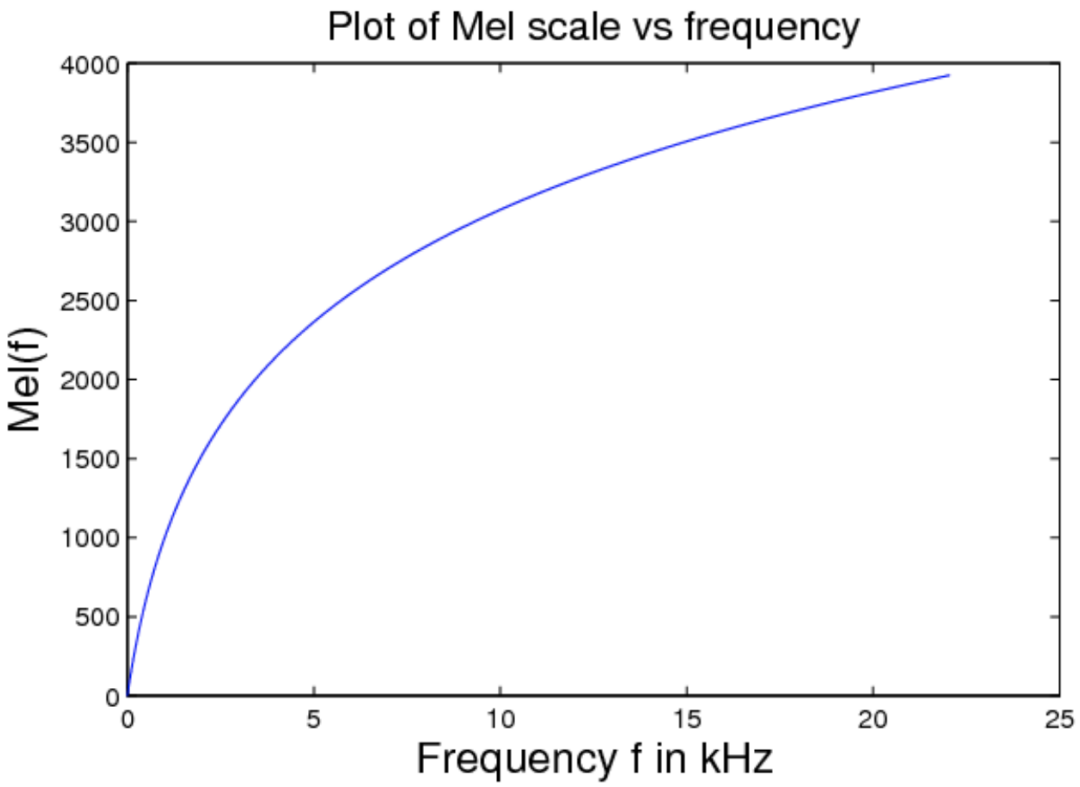

Figure 2.3: Plot showing relationship between mel frequency and actual frequency. *Source: Ramaseshan (2013)*

2.2 Mel Spectrograms

It is often the case that instead of working with audio waveforms directly, they are converted to Mel Spectrograms instead. In this section, we will explain what this conversion consists of and discuss the advantages/disadvantages of using a mel spectrogram representation over the raw waveform. The conversion happens over a series of steps detailed below.

2.2.1 Fourier Transformations

The first step is to split out the wave form into the constituent sinusoidal waves. This transformation is possible owing to Fourier's Theorem (2.2.1). Each sinusoidal wave has an associated frequency and amplitude and the amplitude at each frequency is plotted.

Theorem 2.2.1. If a function is reasonably well-behaved and periodic, then it can be written as a discrete sum of trigonometric or exponential functions with specific frequencies.

Essentially, we have shifted from the time domain to the frequency domain as shown in Figure 2.4

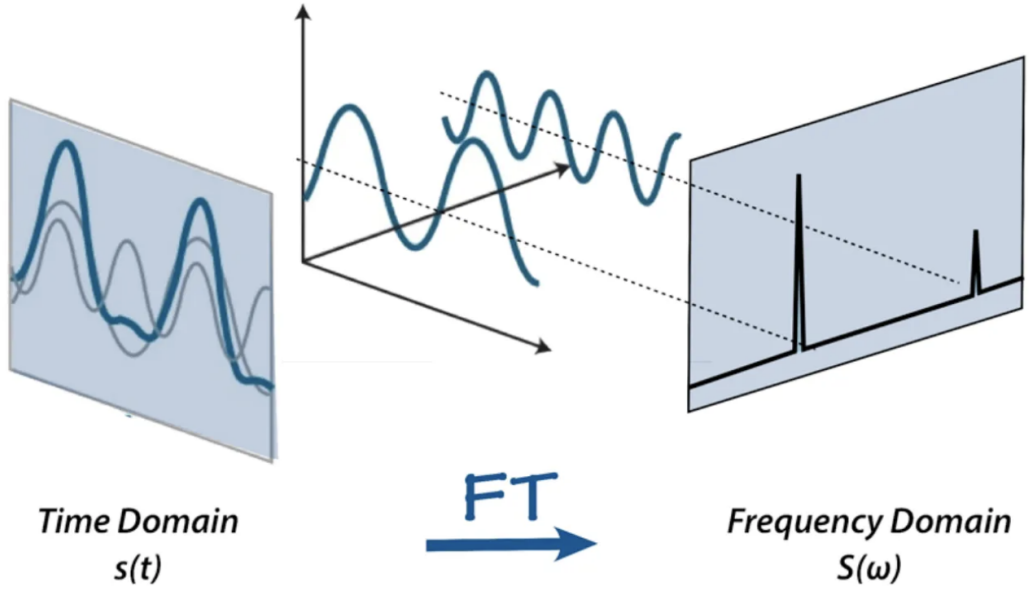

Figure 2.4: Fourier transformation of waveform. *Source: Aavos International*

2.2.2 Spectrogram Generation

As noted in Theorem 2.2.1, the function that we are applying the Fourier transformation to must be periodic. This is unlikely to be the case for most audio. To get around this issue, we apply a technique called short term Fourier transformation ("STFT").

STFT splits the audio into small overlapping windows. The idea is that if the windows are short enough, we can assume that the audio is periodic over this window.

This gives us a graph as in Figure 2.4 for each window. These graphs are then stacked on top of each other, to show how the frequencies change over time, with colour being used to represent the amplitude in units of decibels. The frequencies (y-axis) are also log-scaled. An example spectrogram for speech audio is shown in Figure 2.5 (Left).

2.2.3 Mel-scale Conversion

The final conversion is to convert the y-axis in the spectrogram to the mel scale as per Definition 2.1.5. An example mel spectrogram is shown in Figure 2.5 (Right).

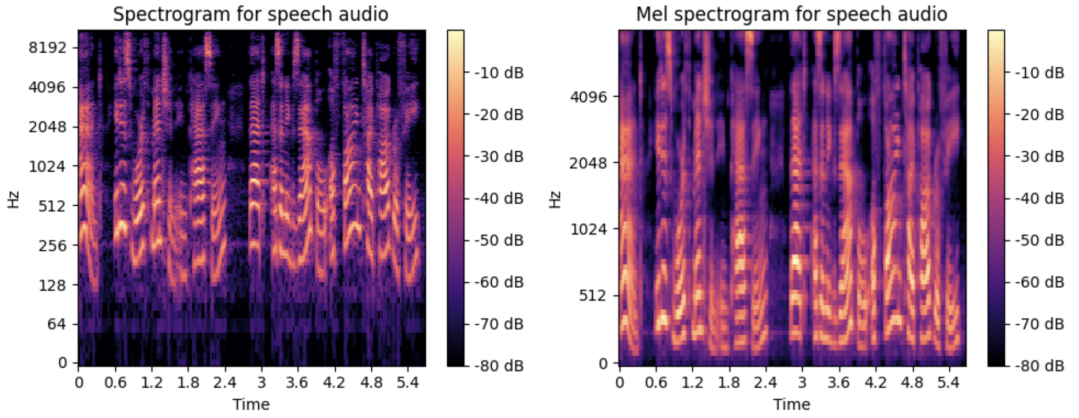

Figure 2.5: Spectrogram and mel spectrogram for the example waveform in Figure 2.1

2.2.4 Advantages and Disadvantages

When a waveform is converted to a mel spectrogram, some level of detail is lost. For example, if we take the waveform shown in Figure 2.1 which is from speech of around 5.5s, at a sample rate of 22050Hz, this has c.125k samples.

When the same waveform is converted to a mel spectrogram, it typically has dimensions of around (80, 245). This loss of information can lead to a degradation of quality.

However the main advantage of using mel spectrograms over the raw waveforms is the reduction of computational power required to predict the mel spectrograms in a text to speech context. In addition, waveforms have a lot more variance making them harder to predict directly. Lastly, most text to speech models employ the use of convolutional layers, and the 2D shape of mel spectrograms makes them well suited for these operations.

In this paper, we want to experiment with a new family of deep learning models, and explore whether these models can be used to directly predict waveforms from text input.

3 Background to Diffusion Models

The goal of this research is to explore the use of cascaded diffusion models for speech generation. Cascaded diffusion models consist of several diffusion models. In this chapter, we will therefore go over the background of diffusion models, provide some intuition as to how they work and discuss how they are combined to form cascaded diffusion models.

3.1 Overview

Diffusion models are state of the art generative models and according to Yang et al. (2022), these models have outperformed Generative Adversarial Networks ("GANs") which have previously dominated the field. Diffusion models were first applied to generate images (Ho et al., 2020), and more recently to generate speech (Kong et al., 2020b).

Diffusion models progressively destroy data by adding noise in a series of steps, usually ending up with Gaussian noise. They then learn to reverse this process in order to generate new samples using a score function. This process is shown diagrammatically in Figure 3.1 below.

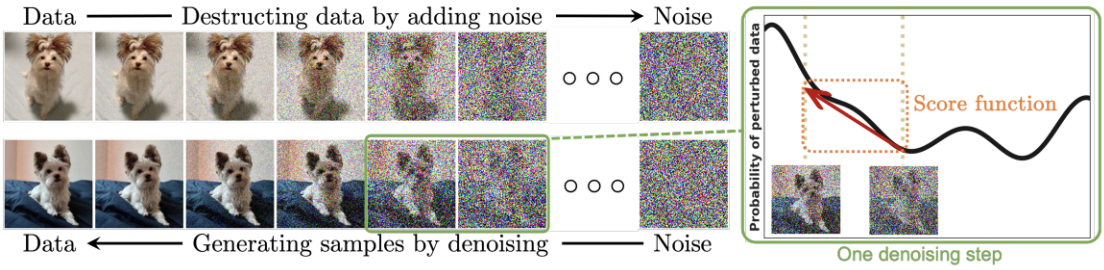

Figure 3.1: Illustration of diffusion process. *Source: Yang et al. (2022)*

The idea behind the models is that the model learns the high level features during the initial stages of the reverse process, and the finer details at the later stages allowing it to learn trajectories to generate different samples.

3.2 Diffusion Model Details

Diffusion models are composed of a forward process, which adds noise to data progressively, and a reverse process, which learns to generate samples from the data space.

3.2.1 Forward Diffusion Process

We start the forward process with the original data sample, x_0, after having chosen the hyper-parameters T and $\beta_1, ..., \beta_T$. T defines the number of times noise is added to data (the number of transition steps), and $\beta_1, ..., \beta_T$ is the noise schedule which determines how much noise is added at each stage. Starting from x_0, the forward process eventually generates x_T which is assumed to be Gaussian noise.

Given data at transition step t-1, x_{t-1}, we make the following assumption about the conditional distribution of $x_t|x_{t-1}$:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_{t-1}\sqrt{1-\beta}_t, \beta_t I) \tag{3.1}$$

The random variables $X_t \sim q(x_t)$ form a Markov chain indexed by $t \in [1, T]$ and therefore we have that:

$$\begin{aligned} q(x_1, ..., x_T|x_0) &= q(x_T|x_0, ..., x_{T-1})q(x_1, ..., x_{T-1}|x_0) \\ &\vdots \\ &= \prod_{t=1}^{T} q(x_t|x_0, ..., x_{t-1}) \\ &= \prod_{t=1}^{T} q(x_t|x_{t-1}) \end{aligned}$$

Where the first equality is through application of Bayes rule, and the last equality is due to the Markov property. If we define $\alpha_t = 1 - \beta_t$ and $\bar{\alpha_t} = \prod_{s=1}^{t} \alpha_s$, then it can be shown that:

$$q(x_t|x_0) = \mathcal{N}(x_0\sqrt{\bar{\alpha}_t}, (1-\bar{\alpha}_t)I) \tag{3.2}$$

where I is the identity matrix.

This gives us a way to obtain x_t directly rather than generating $x_1, ...x_{t-1}$ first as part of the forward diffusion process.

This forward approach introduced by Ho et al. (2020) means that the forward diffusion process does not require any training making it different from other latent variable models such as Variational Auto Encoders ("VAEs").

3.2.2 Reverse Diffusion Process

We will now move on to the reverse diffusion process, which also consists of a Markov chain.

The purpose of the reverse process is to start with a random noise sample x_T, and progressively remove the noise to obtain $x_{T-1}, ..., x_0$ such that x_0 resembles the real data. This reverse process is defined by probabilities:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(\mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \tag{3.3}$$

A neural network is trained to learn the parameters θ in the equation above. Note that in most cases, it is assumed for simplification that $\Sigma_\theta(x_t, t) = \sigma_i^2 I$ where I is again the identity matrix and σ_t^2 is a time dependent constant that is not trained.

Ho et al. (2020) recommend setting $\sigma_t^2 = \beta_t$ or $\sigma_t^2 = \bar{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$

3.2.3 Training

As with VAEs, training is performed by optimising the variational bound on negative log likelihood.

$$\mathbb{E}\left[-\log p_\theta\left(\mathbf{x}_0\right)\right] \leq \mathbb{E}_q\left[-\log \frac{p_\theta\left(\mathbf{x}_{0:T}\right)}{q\left(\mathbf{x}_{1:T} \mid \mathbf{x}_0\right)}\right] =: L$$

$$L = \mathbb{E}_q\left[-\log p\left(\mathbf{x}_T\right) - \sum_{t \geq 1} \log \frac{p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)}{q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right)}\right] \tag{3.4}$$

Ho et al. (2020) show that (3.4) can be written as follows:

$$\begin{aligned} L = \mathbb{E}_q\Bigg[& \underbrace{D_{\mathrm{KL}}\left(q\left(\mathbf{x}_T \mid \mathbf{x}_0\right) \| p\left(\mathbf{x}_T\right)\right)}_{L_T} \\ & + \sum_{t>1} \underbrace{D_{\mathrm{KL}}\left(q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right) \| p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)\right)}_{L_{t-1}} \underbrace{-\log p_\theta\left(\mathbf{x}_0 \mid \mathbf{x}_1\right)}_{L_0} \Bigg] \end{aligned} \tag{3.5}$$

Equation (3.5) calculates the KL divergence between the predicted $p_\theta(x_{t-1}|x_t)$ and the forward posterior:

$$q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right) = \mathcal{N}\left(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t, \mathbf{x}_0\right), \tilde{\beta}_t \mathbf{I}\right) \tag{3.6}$$

$$\text{where} \quad \tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t, \mathbf{x}\right) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}, \mathbf{x}_0 + \frac{\sqrt{\alpha_t}\left(1-\bar{\alpha}_{t-1}\right)}{1-\bar{\alpha}_t}\mathbf{x}_t, \text{ and } \tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t \tag{3.7}$$

This is convenient since we are calculating KL diveregences against Gaussian distributions and so don't need to use Monte Carlo estimates.

Reparameterisation

We note that by fixing $\sigma_t^2 = \beta_t$ or $\sigma_t^2 = \bar{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$, we have that $p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right) = \mathcal{N}\left(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta\left(\mathbf{x}_t, t\right), \sigma_t^2 \mathbf{I}\right)$, and therefore we can write:

$$L_{t-1} = \mathbb{E}_q\left[\frac{1}{2\sigma_t^2}\left\|\tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t, \mathbf{x}_0\right) - \boldsymbol{\mu}_\theta\left(\mathbf{x}_t, t\right)\right\|^2\right] + C \tag{3.8}$$

Here, C is a constant independent of θ and so can be ignored when it comes to optimising the loss function.

If we re-parameterise $\mathbf{x}_t\left(\mathbf{x}_0, \boldsymbol{\epsilon}\right) = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$ for $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and use Equations (3.7), then we obtain:

$$\begin{aligned} L_{t-1} - C &= \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}}\left[\frac{1}{2\sigma_t^2}\left\|\tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t\left(\mathbf{x}_0, \boldsymbol{\epsilon}\right), \frac{1}{\sqrt{\bar{\alpha}_t}}\left(\mathbf{x}_t\left(\mathbf{x}_0, \boldsymbol{\epsilon}\right) - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}\right)\right) - \boldsymbol{\mu}_\theta\left(\mathbf{x}_t\left(\mathbf{x}_0, \boldsymbol{\epsilon}\right), t\right)\right\|^2\right] \\ &= \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}}\left[\frac{1}{2\sigma_t^2}\left\|\frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t\left(\mathbf{x}_0, \boldsymbol{\epsilon}\right) - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}\right) - \boldsymbol{\mu}_\theta\left(\mathbf{x}_t\left(\mathbf{x}_0, \boldsymbol{\epsilon}\right), t\right)\right\|^2\right] \end{aligned} \tag{3.9}$$

One option is therefore to train a model μ_θ that takes $\mathbf{x}_t$ as an input and tries to predict $\tilde{\boldsymbol{\mu}}_t$, however we can re-parameterise a second time using

$$\begin{aligned}\boldsymbol{\mu}_\theta\left(\mathbf{x}_t, t\right) &= \tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t, \frac{1}{\sqrt{\bar{\alpha}_t}}\left(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_\theta\left(\mathbf{x}_t\right)\right)\right) \\ &= \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta\left(\mathbf{x}_t, t\right)\right)\end{aligned}$$

Now, our model is a function $\boldsymbol{\epsilon}_\theta$ which predicts $\boldsymbol{\epsilon}$ given $\mathbf{x}_t$ and t. This simplifies Equation (3.9) to:

$$L_{t-1} - C = \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{\beta_t^2}{2\sigma_t^2\alpha_t\left(1-\bar{\alpha}_t\right)}\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t\right)\right\|^2\right] \tag{3.10}$$

Ho et al. (2020) claim that training a model $\boldsymbol{\epsilon}_\theta$ to predict $\boldsymbol{\epsilon}$ yields better results than training a model μ_θ to predict $\tilde{\boldsymbol{\mu}}_t$ according to their experiments on images.

Training Algorithm

Using the re-parameterisation in the previous section, we can now discuss the training algorithm. The key point is that the funciton we are trying to estimate, $\boldsymbol{\epsilon}_\theta$, depends on both $\mathbf{x}_0$ and t. Our training set therefore essentially becomes all combinations of $\mathbf{x}_0$ and t. Ho et al. (2020) propose an algorithm where the $\mathbf{x}_0$ and t are chosen randomly and separately using a uniform distribution.

For each combination of $\mathbf{x}_0$ and t, we sample $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$ and obtain $\mathbf{x}_t$ using Equation (3.2):

$$x_t = x_0\sqrt{\bar{\alpha}_t} + \boldsymbol{\epsilon}\sqrt{(1-\bar{\alpha}_t)}$$

The neural network then makes a prediction for $\boldsymbol{\epsilon}$ and model parameters are updated via gradient descent. This algorithm is summarised in Algorithm 1.

Algorithm 1 Diffusion model training

1: **repeat**
2: Sample $\mathbf{x}_0 \sim q_{\mathbf{x}_0}$
3: Sample $t \sim Uniform(1, ..., T)$
4: Sample $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$
5: Predict $\boldsymbol{\epsilon}' = \boldsymbol{\epsilon}_\theta(x_0\sqrt{\bar{\alpha}_t} + \boldsymbol{\epsilon}\sqrt{(1-\bar{\alpha}_t)}, t)$
6: Update θ based on $\nabla_\theta \left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}'\right\|^2$
7: **until** Convergence

3.2.4 Sampling

Sampling is the process of generating new samples from a trained diffusion model. We start the process with Gaussian noise $\mathbf{x}_T$ and use our trained model to obtain $\mathbf{x}_{T-1}, ..., \mathbf{x}_0$.

More precisely, given $\mathbf{x}_t$, we compute $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta \left(\mathbf{x}_t, t \right) \right) + \sigma_t \mathbf{z}$, where $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

When we reach $t = 1$, we no longer add further noise and $\mathbf{x}_0 = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_1 - \frac{\beta_1}{\sqrt{1-\bar{\alpha}_1}} \boldsymbol{\epsilon}_\theta \left(\mathbf{x}_1, 1 \right) \right)$.

This process is summarised in Algorithm 2.

Algorithm 2 Diffusion model sampling

1: Sample $\mathbf{x}_T \sim \mathcal{N}(0, I)$
2: **for** t=T,...1 **do**
3: Sample $\mathbf{z} \sim \mathcal{N}(0, I)$ if $t > 1$ else $\mathbf{z} = 0$
4: Compute $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta \left(\mathbf{x}_t, t \right) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

3.2.5 Conditioning

A key aspect of text to speech synthesis is conditioning the model. This simply means that our model $\boldsymbol{\epsilon}_\theta$, takes an additional conditioning parameter $\mathbf{h}$ which guides the model. This parameter tells the model what kind of data to generate. In the context of text to speech models, this conditioning parameter may be the text itself (or an embedding of this text) and in the case of vocoders this is usually the mel spectrogram.

3.3 Network Architecture: U-Net

The most common neural network architecture used in diffusion models is a U-Net based architecture (Ronneberger et al., 2015). U-Nets were initially introduced for biomedical image segmentation and work well with problems aimed at obtaining predictions for each pixel or groups of pixels in an image. This is because the input and output dimensions are similar if not the same. For example in biomedical image segmentation, each area of an image or pixel may need to be evaluated to predict risk factors.

Figure 3.2 highlights the architecture used by Ronneberger et al. (2015) for biomedical image segmentation. The architecture consists of a contracting path (left) followed by an expansive path (right). The contracting path reduces the dimensions of the image whilst increasing the number of channels and the expansive path reverses this process. This forms a U-shaped network which gives the model its name.

The contracting path consists of four blocks and each block consists of a sequence of three convolutional layers with a 3×3 kernel. The first and second convolutional layers in each block are followed by rectified linear unit ("ReLU") activation layers and the second convolutional layer in each block doubles the channel size. A copy of the resulting tensor is retained and passed on to the associated block in the expansive path as shown in Figure 3.2. This is referred to as a skip connection. After each block, a 2×2 max pooling layer is used with a stride of 2 to halve the dimensions of the image.

The model contains a block in the middle (with a similar structure to the blocks in the contracting path) before the expansive path begins. Each block in the expansive path is first preceded by a 2×2 transposed convolutional layer. The blocks in the expansive path themselves have the same structure as the blocks in the contracting path. The output is concatenated with the skip connection from the associated block from the contracting path along the channel dimension.

These types of networks are effective because the contracting path captures the context along the channel dimension, which is then localised by the expansive path. Moreover, deep neural networks can suffer from several issues such as degradation or vanishing gradients. Skip connections are one way of reducing these issues and have been used in other types of models such as long short-term memory ("LSTM") models (Hochreiter and Schmidhuber, 1997).

According to Salimans et al. (2017), down-sampling increases the receptive field of the model (in the same way as dilated convolutions [1] but in a more computationally efficient manner) whilst the skip connections prevent loss of information from the down-sampling.

3.3.1 U-Nets in diffusion models

U-Nets are highly suitable for diffusion models since the input to the model (noise) is the same size as the output and has the same number of channels. However, some adjustments are made to the architecture to allow them to be used for this purpose. For example, the convolutional layers in each block use zero-padding to ensure the image size is consistent and only changes when the max pooling or transposed convolution layers are applied. This simplifies the architecture as the skip connections no longer need to be cropped as the contracting and expansive paths are symmetric in nature.

In addition, Ho et al. (2020) recommend the use of group normalisation in the U-Nets to accelerate training and Salimans et al. (2017) experimented with the use of dropout in similar networks and recommended its use to prevent overfitting.

[1] These are convolutional layers that skip inputs from the previous layer with a given step size. See Section 4.1.1 for further details.

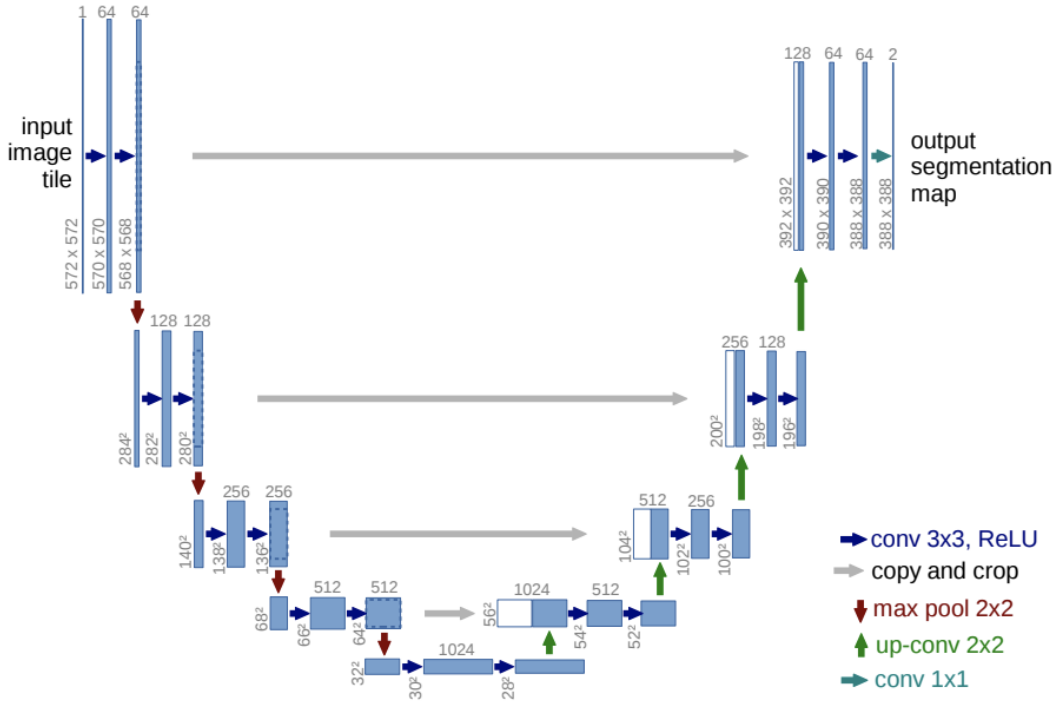

Figure 3.2: U-Net model architecture. Each blue box corresponds to a multi-channel feature map with the number of channels denoted above each box (also represented by the width of the boxes). The dimensions of the images are provided below each box and are represented by the height of the boxes. White boxes represent copied feature maps from skip connections. The arrows denote the different operations. *Source: Ronneberger et al. (2015)*

3.4 Cascaded Diffusion Models

A recent advancement in the field of diffusion models has been the introduction of cascaded diffusion models ("CDMs") by Ho et al. (2022) which boast superior performance over other methods such as GANs and standard diffusion models when it comes to image generation according to the authors.

A CDM comprises of a pipeline of multiple diffusion models that generate samples of increasing resolution. The CDM starts with a standard diffusion model at the lowest resolution, followed by a series of super-resolution diffusion models that successively upsample the image and add higher resolution details. This is illustrated in Figure 3.3 using the example of image generation.

The idea is that the first model learns the high level details related to the space of the data whilst the subsequent super resolution models learn the finer details. The idea of

applying CDMs can be extended to audio which is the core part of our research. In the case of audio, we aim to use a basic diffusion model to generate samples with low sample rates and use successive super resolution models to increase the sample rate of the output.

-

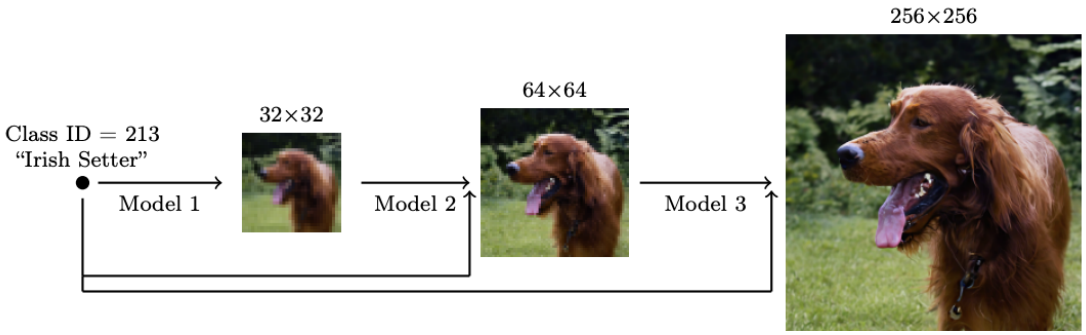

Figure 3.3: Illustration of image generation using a cascaded diffusion model. *Source: Ho et al. (2022)*

Ho et al. (2022) utilise the Super Resolution 3 ("SR3") model (Saharia et al., 2022) to increase the resolution of the images. This is a U-Net based diffusion model that is conditioned on the output of the previous model in the pipeline.

The first model, ϵ_0 takes input parameters $\mathbf{x}_t^0$, t, and any conditioners $\mathbf{h}$ as before to estimate $\mathbf{x}_0^0$ at the base resolution. We use the notation $\mathbf{x}_t^i$ to represent the data at transition step t, at the ith resolution in the pipeline.

The resolution of $\mathbf{x}_0^0$ is increased using methods such as cubic interpolation to obtain $\mathbf{x}_0^{1'}$ which can be thought of as a quick approximation of the data at the higher resolution. The next model in the pipeline, ϵ_1 then takes parameters $\mathbf{x}_t^1$, t, and $\mathbf{h}$ at each step of diffusion but also takes predictions from the previous model; $\mathbf{x}_0^{1'}$. Ho et al. (2022) and Saharia et al. (2022) recommend the use of a similar structure for the super resolution models as the base diffusion models, and to simply concatenate the inputs $\mathbf{x}_0^{1'}$ and $\mathbf{x}_t^1$ along the channel dimensions. In other words, the super resolution models perform the same task, but with extra information from the previous model in the pipeline.

This process is repeated based on the number of super resolution models used. Both Ho et al. (2022) and Saharia et al. (2022) use increasingly larger models in the pipeline to account for the increase in resolution.

An important point to note is that each model is trained in succession by using the outputs from the previous model. However Ho et al. (2022) also recommend the use of conditional augmentation, where noise is added to the outputs of one model, before being used to train the net model.

This can be done in one of two ways. The first is to choose a hyper-parameter σ and add Gaussian noise centred at 0 with standard deviation σ to $\mathbf{x}_0^0$ before the interpolation takes place. The second solution is to use the forward diffusion process from the first model and obtain $\mathbf{x}_s^0$ from $\mathbf{x}_0^0$. Now, our hyper-parameter of choice becomes $s \in [0, T]$. The idea is that each model can learn to correct for the noise in the outputs of the previous model. Interestingly, during inference conditional augmentation is not used.

Whilst Ho et al. (2022) reported conditional augmentation to have a large and positive impact on quality, we will not be exploring this in our thesis as it will not be feasible to train multiple diffusion models and find the optimal values for σ or s. In other words, we will set $\sigma = 0$ or equivalently $s = 0$. This brings us on to one of the downsides of diffusion models and indeed cascaded diffusion models in particular; it can take a lot of time to tune the hyper-parameters and find the optimal combination.

4 Recent Developments in the Field of Speech Synthesis

In this chapter, we will discuss some of the recent literature in the domain of speech generation, and explain how our proposed solution solves some of the shortcomings of these approaches.

4.1 Autoregressive Models

Deep learning approaches to speech generation only began development around 2016 and immediately proved to be very successful. The first group of models were autoregressive models which utilise the sequential nature of waveforms in their predictions.

The key component of autoregressive models is that given a waveform $\mathbf{x} = (x_1, ..., x_L)$ of length L, the joint probability can be factorised as a product of conditional probabilities as follows:

$$p(\mathbf{x}) = \prod_{t=1}^{T} p\left(x_t \mid x_1, \ldots, x_{t-1}\right) \tag{4.1}$$

In other words, for any $l \in 1, ..., L$, x_l is conditioned on all the samples before it $(x_1, ..., x_{l-1})$.

One of the biggest issues with autoregressive models is that whilst training can be quick, inference is extremely slow since each sample needs to be generated in turn.

4.1.1 WaveNet (Oord et al., 2016)

One of the earliest deep learning models used in speech synthesis was WaveNet (Oord et al., 2016) which is an autoregressive model for generating waveforms.

The main ingredient of the WaveNet architecture is the use of causal convolutional layers which is illustrated in Figure 4.1. These layers aren't fully connected, and each node in position i of layer j only receives inputs from nodes i-1 and i of layer j-1.

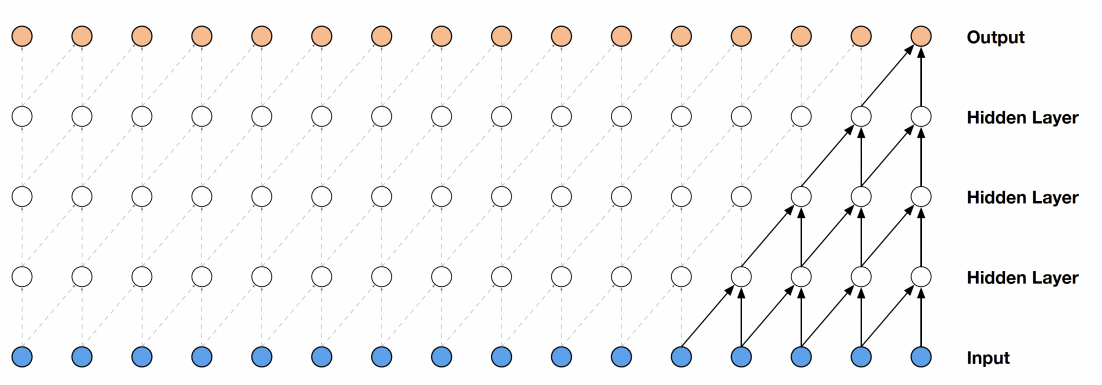

Figure 4.1: Illustration of a stack of causal convolutional layers. *Source: Oord et al. (2016)*

This allows the network to achieve the autoregressive property. However as can be seen from Figure 4.1, this means that the probability of a sample at position $l \in 1, ..., L$ is only conditioned on $x_{l-1}, ..., x_{l-d}$ where d is the number of layers.

Given that most waveforms will have a sample rate of at least 16kHz, we would need to stack a huge number of layers in order to condition on a set of samples with sufficient look back. This would cause the network to be very inefficient.

To get around this issue, Oord et al. (2016) introduced dilation to the causal convolutional layers as shown in Figure 4.2. Each layer has an additional property called dilation, which causes input values to be skipped with a certain step. This allows the network to have a wider field of vision, similar to having strides except the output size is maintained.

This is an important concept and is used in many subsequent models discussed below as it allows the receptive field to be increased without adding a large number of layers. The downside to this approach is that if the dilation in each layer is too high, the receptive field can be large but the model will be too coarse and contain blind spots due to the large portions of the sequence that are skipped.

Finally, WaveNets are conditioned on text so that the predicted speech audio is based on a text input. This results in Equation 4.1 becoming:

$$p(\mathbf{x}) = \prod_{t=1}^{T} p\left(x_t \mid x_1, \ldots, x_{t-1}, \mathbf{h}\right)$$

where the vector $\mathbf{h}$ contains global features that are independent of the position such as speaker prosody and local features.

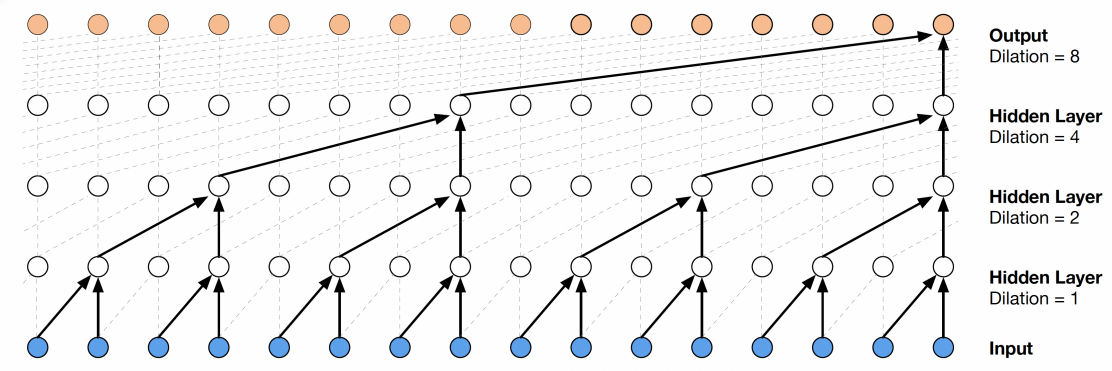

Figure 4.2: Illustration of a stack of dilated causal convolutional layers. *Source: Oord et al. (2016)*

The local features used to condition WaveNets are extensive and include phonemes, phoneme durations, log fundamental frequencies, syllables per word, syllable stresses. Whilst this helps the model start off with a good context of the speech, producing these features requires extensive domain knowledge and the training of several separate models which limits the usability of WaveNet in practice.

4.1.2 Tacotron

Tacotron (Wang et al., 2017) is a text to speech model that uses text input to generate mel spectrograms before converting these mel spectrograms to waveforms. It addresses the complexities in the inputs to WaveNet by training the model to learn the linguistic features such as syllable stretches etc. rather than explicitly providing these to the model.

If we consider a mel spectrogram, it is easy to calculate the number of samples in the associated waveform by using the properties of the mel spectrogram and the sample rate of the waveform. However there is no direct link between the length of a text sequence and the number of frames in a mel spectrogram or the samples in a waveform.

To get around this issue, Tacotron uses sequence to sequence modelling (Sutskever et al., 2014) which consists of using two long short-term memory ("LSTM") based models. The first LSTM maps the input text to a vector of fixed dimension whilst the second LSTM maps the fixed dimension vectors to mel spectrograms. This approach is used in situations where the input and output sequences can vary in length for example in language translation tasks.

Once the mel spectrogram is generated, it is converted to waveform using the Griffin-Lim algorithm (Lim, 1984). This algorithm is a not a deep learning based method to generate waveforms and provides a good estimate. However it does not perform as well

as deep learning based methods as shown by (Shen et al., 2018).

Tacotron2 (Shen et al., 2018) uses a similar approach to Tacotron to generate mel spectrograms, but uses a modified version of WaveNet that is conditioned on mel spectrograms instead of the array of linguistic features used by WaveNets. Shen et al. (2018) reported that this approach provided significantly better performance over Tacotron and WaveNet in terms of human opinion scores.

Tacotron2 addresses the deficiencies in the vocoder used in Tacotron and simplifies the inputs required to condition the generation as compared to WaveNet. However, the vocoder is still an autoregressive model and therefore struggles with poor inference times.

4.1.3 Transformer-TTS (Li et al., 2019)

We will now discuss the final model in the auto-regressive family of models.

Transformer-TTS (Li et al., 2019) is similar to Tacotron2 in that it has two elements: a mel spectrogram generator; and WaveNet conditioned on these mel spectrograms. The key difference is that the RNN's in the mel spectrogram generator are replaced with transformers (Vaswani et al., 2017).

One of the biggest issues with Tacotron(2) is that the RNNs can only generate the outputs sequentially. This is because the previous hidden state and current input are both required to generate the current output. As a result, it is difficult to parallelise the models during training and inference.

The use of transfomers fixes this issue, and Li et al. (2019) claim that Transformer-TTS is able to achieve a similar level of performance to Tacotron2, but the training time is reduced by a factor of c.4.27.

Note that there many other autoregressive speech synthesis models which we haven't covered in this paper. There have also been a number of flow based models such as Glow-TTS (Kim et al., 2020) which have attempted to solve the issue of long inference times in autoregressive models. However these models did not perform as well as the models we will consider in the next couple of sections.

4.2 Generative Adverserial Networks

With the introduction of Generative Adverserial Networks ("GANs") (Goodfellow et al., 2014), a new family of models emerged which were hugely successful for image generating tasks. Similar to VAEs, GANs also learn to create mappings from a space $\mathcal{Z}$ of low-dimensional latent vectors with a known prior distribution P_Z to a space $\mathcal{X}$ of real data.

GANs consists of two elements: the generator, G; and the discriminator, D. The generator maps data from the latent space to the space of real data ($G : \mathcal{Z} \mapsto \mathcal{X}$). The discriminator tries to differentiate the real data from generated data ($D : \mathcal{X} \mapsto \{0, 1\}$).

The generator is trained using the following loss function:

$$L_G = \mathbb{E}_{\boldsymbol{x} \sim P_X}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim P_Z}[\log(1 - D(G(\boldsymbol{z})))]$$

The discriminator is trained on the negative of the above loss function ($L_D = -L_G$). In other words, the generator and discriminator are set against each other and the idea is that they force each other to improve [1].

In this section we will describe some of the applications of GANs in the field of speech synthesis.

4.2.1 WaveGAN and SpecGAN (Donahue et al., 2018)

Radford et al. (2015) introduced deep convolutional GANs ("DCGANs") which utilise convolutional layers in both the generator and discriminator to generate images. The generator uses a series of 2-dimensional transposed convolutional layers to up-sample data until a high resolution image is generated. The discriminator uses a series of 2-dimensional convolutional layers with strides to down-sample images and create a binary prediction.

Radford et al. (2015) showed promising results for image generation using DCGANs and Donahue et al. (2018) adapted them to create the earliest unsupervised speech generation models utilising GANs; WaveGAN and SpecGAN. WaveGAN generates waveforms directly while SpecGAN generates mel spectrograms.

In order to use DCGANs for waveforms in WaveGAN, Donahue et al. (2018) noted that it was important to allow the convolutional layers to have a large field of vision. They therefore adopted the 5×5 filters in DCGAN to 1-dimensional filters with a size of 25. In addition, due to the large number of samples that need to be generated for direct waveform prediction, additional transposed convolutional layers were stacked and the upsampling rate was doubled.

For their spectrogram generating model SpecGAN, Donahue et al. (2018) took fixed length samples from the audio and converted them to mel spectrograms ensuring the number of frequency channels equalled the number of frames. In other words, the 2-dimensional arrays representing the mel spectrograms had the same number of rows

[1] Note that we are simplifying the loss functions as GANs are not the primary focus of this thesis. In practice, more complex and non-symmetric loss functions are used that use a least squares based approach rather than the binary cross-entropy approach shown above in order to avoid vanishing gradient flows. Interested readers are invited to read (Mao et al., 2017) for further details.

as columns. This meant that the DCGAN approach used for images could be directly applied to generate mel spectrograms.

Whilst Donahue et al. (2018) showed that it was possible to adopt GANs to generate waveforms or mel spectrograms, the quality of the outputs in terms of the MOS was low when compared to the MOS of the ground truth audio. Surprisingly, the direct waveform generating model performed better although this may be due to the fact that the outputs from SpecGAN were converted to waveforms using the Griffin-Lim algorithm (Lim, 1984) which is an outdated approach that under-performs when compared to more recent deep learning approaches (Shen et al., 2018).

4.2.2 HiFi-GAN (Kong et al., 2020a)

HiFi-GAN is a groundbreaking vocoder that generates waveforms of speech using their mel spectrograms. It achieves a similar level of quality to the state of the art autoregressive models discussed in Section 4.1 without the issue of extremely long inference times even when generating audio in high resolutions (22,050Hz). Due to it's success, some subsequent text to speech models such as Grad-TTS (Popov et al., 2021) only deal with the task of generating mel spectrograms from text, and pair their model with HiGi-GAN for an end to end solution.

Generator

The generator in HiFi-GAN uses a series of transposed convolutional layers to up-sample the mel spectrograms so that they have the same dimension as the audio being generated, using an approach similar to the one introduced by Donahue et al. (2018). However Kong et al. (2020a) introduce elements called Multi-receptive field fusion ("MRF") after each transposed convolutional layer.

Each MRF element consists of a series of residual blocks that contain dilated convolutional layers with different kernel sizes and dilations as well as a leaky ReLU activation layer. The output from a transposed convolutional layer passes through each of the residual blocks, and the outputs from these blocks are summed. The idea is that each residual layer has a different receptive field based on the kernel size and dilation, and summing all the results gives the model the ability to have a diverse receptive range and ability to pick up on patterns at varying granularities.

Discriminator

Kong et al. (2020a) noted that the discriminator needed to be able to identify diverse periodic patterns and the simple approach applied by Donahue et al. (2018) wasn't sufficient in doing so. Kong et al. (2020a) therefore developed an approach using a

series of sub-discriminators. Each sub-discriminator takes samples from the audio being assessed at fixed time intervals. A final sub-discriminator takes a more holistic view.

Each sub-discriminator has it's own loss function, and the overall discriminator loss is the sum of the losses from the sub-discriminators. This essentially means that the generator is trained to "fool" an array of discriminators that assess the audio through different lenses.

4.3 Diffusion Models

Similar to image generation, GAN's held the lead in terms of the quality of speech generated and usability. However, in the last couple of years, researchers have adopted the latest diffusion based techniques used for image generation and shown that they can be used for speech generation as well. In this section we will discuss some of these models.

4.3.1 DiffWave (Kong et al., 2020b)

One of the earliest implementations of diffusion models in the audio domain is the DiffWave vocoder (Kong et al., 2020b) which transforms mel spectrograms into speech waveforms. This model also achieved the quality of autoregressive models with reasonable inference times, however the authors did not compare the model performance to HiFi-GAN and so it is difficult to compare the two approaches.

DiffWave uses the approach described in Algorithms 1 and 2 to train the model and sample at each stage of the diffusion process. As well as the diffusion timestep t, the trained model is conditioned on the mel spectrogram associated with the speech waveform. This allows the model to transform the white noise to waveforms given the context of the mel spectrogram.

Despite having the same training/sampling algorithms and loss functions as the ones recommended by Ho et al. (2020), DiffWave uses a different architecture instead of a U-Net based architecture.

Interestingly, although DiffWave has a very different training process and loss function, it takes a similar approach to HiFi-GAN. The idea, as before, is to be able to capture the periodic information at different granularities simultaneously. Therefore DiffWave consists of a series of residual blocks, which have different dilations (but the same kernel sizes). There are however two key differences in the structures of HiFi-GAN and DiffWave.

The first difference is that HiFi-GAN takes the mel spectrogram and alternates between applying transposed convolutional layers and the MRF elements. On the other hand,

DiffWave uses transposed convolutional layers on the mel spectrograms to match the dimensions of the target waveform before any dilated convolutional layers are applied. Once the mel spectrogram and waveform have the same dimensions, a series of residual blocks is applied.

The second difference is that HiFi-GAN applies the residual blocks in each MRF in parallel and sums the results after each transposed convolution. This approach would not work in the DiffWave architecture since there is only one equivalent of the MRF. This would mean all of the residual blocks are applied in parallel, severely restricting the receptive field. Instead the approach taken in DiffWave is to apply each residual block in succession, but creating skip connections in between each residual blocks so that the skip connections can be summed at the end. Since the number of residual blocks each skip connection has been through will be different, this allows DiffWave to observe different periodic patterns together whilst achieving a large receptive field (since some of the connections will have gone through multiple residual blocks).

One departure from the diffusion approach used by Ho et al. (2020) is the use of a fast sampling approach. Kong et al. (2020b) noted that the most important diffusion steps were those closer to 0. They therefore found that the reverse process can be collapsed into less steps, with an adjusted noise schedule that is determined by the user.

More concretely, suppose a user provides an inference schedule $\{\eta_t\}_{t=1}^{T_{\text{infer}}}$ during inference where $T_{infer} << T$. Then in a similar fashion to before, we define $\gamma_t = 1 - \eta_t$ and $\bar{\gamma}_t = \prod_{s=1}^{t} \gamma_s$. Kong et al. (2020b) show that we can infer the value of t by setting $t_s^{\text{align}} = t + \frac{\sqrt{\bar{\alpha}_t} - \sqrt{\bar{\gamma}_s}}{\sqrt{\bar{\alpha}_t} - \sqrt{\bar{\alpha}_{t+1}}}$ if $\sqrt{\bar{\gamma}_s} \in [\sqrt{\bar{\alpha}_{t+1}}, \sqrt{\bar{\alpha}_t}]$. Here, $\bar{\alpha}_t$ is as previously defined in Section 3.2.1. We can then apply the sampling process from Algorithm 2 using this new noise schedule and t_s^{align} .

We will explore the exact structure of DiffWave in Section 5.3 where we discuss how we have adapted the model for our experiments.

4.3.2 Grad-TTS (Popov et al., 2021)

Up until now, and indeed for the majority of this thesis, we assume that the random variables X_t of data samples at transition step t of the forward diffusion process form a Markov Chain (See Section 3.2.1). However Song et al. (2020) provide an alternative approach which assumes that X_t instead follows a stochastic differential equation. In this thesis, we will only cover the final implications of this approach but interested readers are invited to explore (Song et al., 2020).

Recall that we previously assumed $X_T \sim \mathcal{N}(0, \Sigma)$. The first difference caused by this approach is that $X_T \sim \mathcal{N}(\mu, \Sigma)$ where μ can be chosen and Σ is usually assumed to be the identity matrix I for simplicity. We will discuss the implications of this generalisation on the diffusion process further on.

Song et al. (2020) show that under this approach, the reverse diffusion process becomes an ordinary differential equation ("ODE"):

$$\frac{dX_t}{dt} = \frac{1}{2}\left(\left(\mu - X_t\right) - \nabla \log p_t\left(X_t\right)\right)\beta_t \tag{4.2}$$

At each timestep t, we estimate the above ODE and solve backwards to get to X_{t-1}. In order to estimate the ODE, a neural network is trained to predict the gradient of the log-density of noisy data $\nabla \log p_t\left(X_t\right)$.

Popov et al. (2021) used this approach to create Grad-TTS which generates mel spectrograms from text inputs. Grad-TTS uses a text encoder similar to the one used by Li et al. (2019), however this encoder is trained to predict $\mu = x_0$ as well as possible. x_T is then sampled from $\mathcal{N}(\mu, \Sigma)$ and a U-Net predicts $\nabla \log p_t\left(X_T\right)$ to solve the ODE in Equation 4.2 and obtain x_{T-1}. The process is repeated to obtain x_0.

The U-Net takes as input x_t, t, and μ. Popov et al. (2021) claim that since the encoder tries to get μ as close to x_0 as possible, the starting point of the reverse diffusion process is already closer to x_0 than under the standard Markov chain approach which helps improve the model performance.

Another benefit of this approach is that when the ODE in Equation 4.2 is estimated by a trained model, instead of going back one transition step during the reverse diffusion process, the gradient can be used to take a number of steps at once. This would clearly cause the inference to be more coarse and the quality to be reduced (this is similar to using a larger learning rate during training), but it would help speed up the inference time. The nice thing about this approach is that the model can be trained on an infinite number of steps $t \in [0, 1]$ and at inference the user can decide how many steps to take during the reverse process. This gives the user the flexibility to choose between quality and inference speed without re-training the model.

Popov et al. (2021) pair the Grad-TTS model with HiFi-GAN to create an end to end text to speech model which according to the results of the authors surpasses all other end to end text to speech models in terms of the mean opinion score.

4.4 Potential Impact of using Cascaded Diffusion Models

Given the recent developments in the field of speech synthesis and in particular the research into speech generating diffusion models, the natural question to ask is where CDM's fit into this landscape.

In our opinion, diffusion models have shown a lot of promise as shown above, however there is potential to further improve these models and adapt the recent research in image generation. In addition, current diffusion models don't cover the whole end to end text

to speech problem on their own, and whilst they can be paired with other models to complete the task, there has not been any research to our knowledge in optimising these models to work together.

Using CDM's not only allows us to create a complete text to speech model as per our research goal, it can also help increase the accuracy of the approach due to the ability for the models to be tuned together using techniques such as conditional augmentation (see Section 3.4). This helps us investigate our first research sub-goal.

Moreover, models such as Grad-TTS take a long time to train, and by using CDMs we would like to research the implications of using lower resolutions for the base model. For example, if a smaller base model is used to generate mel spectrograms at lower resolutions, would we still be able to generate high resolution waveforms using a CDM pipeline. The reason this question is relevant is because the base model is the model that learns the prosody, and general aspects of the speech. This model therefore has a big impact on the nature of the speech output at the end, therefore being able to use a smaller model for this task would facilitate tuning to not just optimise the hyper-parameters but to tailor the base model to different speakers/datasets.

5 Methodology

In this chapter, we will explain the building blocks of our models, and discuss their configuration in detail. In the following chapter, we will discuss how these building blocks are brought together for the various experiments. It should be noted that some of the building blocks may not be used in particular experiments. For example in some experiments we deal with using CDMs as vocoders and thus any elements related to processing text would not be used for these experiments.

5.1 Text Encoder

The first task in a text to speech model is to embed the text into a format that can be used by the base model as a conditioner effectively. This task is achieved by the encoder. The input to the encoder is the text after being converted to a sequence of phonemes and the output is a sequence of embeddings that can be used by the model. We use the Carnegie Mellon University Pronouncing Dictionary ("CMU dictionary") (Weide et al., 2015) to convert the text into a sequence of phonemes.

This approach is intuitive because the input to each model is how we expect those words to be pronounced. Note that we could have also trained the encoder to predict the phonemes, or use a pre-net to derive the phonemes, however there is general consensus as to the way words should be broken down into the constituent phonemes and so we use a dictionary to simplify the process. This technique is commonly used in text to speech models for example (Wang et al., 2017) and (Li et al., 2019).

There are limitations to this approach due to the fact that the pronounciation of a word is not determined by the context in which the word is used. For example, some words in the English language are heteronyms (also known as heterophenes) which are pairs or triplets of words that have the same spelling but different pronunciations and meanings. For example the word "lead" has completely different pronunciations when used to refer to the metal as opposed to when it is used to refer to the verb. That being, said, a lookup in a phoneme dictionary should be sufficient for the majority of cases.

We use a similar approach for the encoder as Li et al. (2019). The encoder architecture is summarised in 5.1 and consists of a pre-net and N residual blocks.

The first part of the encoder is a pre-net which consists of an embedding layer, a fully

connected layer followed by the ReLU activation function.

The number of residual blocks in the encoder is a hyper-parameter that can be tuned. Each residual block contains a multihead attention layer, followed by a feed forward network ("FFN"). The residual blocks also contain ReLu activation layers followed by layer normalisation. This is a pretty standard architecture that is used in transformer models, the only difference is that it is paired with a different decoder.

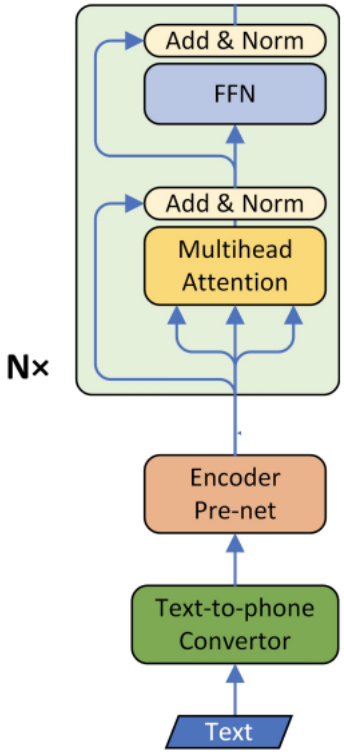

Figure 5.1: Encoder architecture. *Source: Li et al. (2019)*

5.2 Duration Predictor

The task of the duration predictor is to predict the duration of each phoneme in the provided sequence. In terms of the architecture, this is simply two fully connected layers with a sigmoid activation function that predicts the log durations for each phoneme in the sequence.

However the difficulty arises in getting the ground-truth durations for each phoneme since these are also unknown but required to train the model. In the following section,

we will describe how these durations are estimated using two approaches.

5.2.1 Monotonic Alignment Search

Kim et al. (2020) propose to estimate the durations of phonemes using Monotonic Alignment Search ("MAS") which utilises dynamic programming and we have attempted to adopt a similar approach.

Let $\mathbf{h} = \mu_1, ..., \mu_{T_{text}}$ be our phoneme sequence embeddings where T_{text} is the total length of the phoneme sequence.

Let $\mathbf{z} = z_1, ..., z_{T_{wave}}$ be the latent representation our waveform where T_{wave} is the total number of samples in the waveform.

Suppose that all the values in the latent variable are normally distributed, that is $z_j \sim \mathcal{N}(\mu_j, \sigma_j)$. Note that in our case we assume $\sigma_j = 1 \forall j$ for simplicity and our encoder only outputs the means.

Definition 5.2.1. An **alignment function A** is a surjective and monotonic function from the indices of the latent variables **z** (1,...,T_{wave}) to the indices of the phoneme embeddings **h** (1,...,T_{text}). In other words:

$$A : \{1, ..., T_{wave}\} \mapsto \{1, ..., T_{text}\}$$

$$\text{such that } A(j) \geq A(j') \quad \forall j \geq j'$$

$$\text{and } \forall i \in \{1, ..., T_{text}\} \quad \exists j \in \{1, ..., T_{wave}\} \quad \text{such that } A(j) = i$$

Figure 5.2 (Left) shows an example of alignment between the embedding vector **h** and the latent representation **z**.

Our goal is to find the optimal Alignment A^* which maximises the log-likelihood

$$\sum_{j=1}^{T_{wave}} \log\left(\mathbb{P}_{\mathcal{N}(\mu_{A(j)},1)}(z_j)\right)$$

The first step of MAS is to create a matrix Q where $Q_{i,j}$ is the maximum log-likelihood of any alignment between the first j indices of **z** and the first i indices of **h**.

Due to the surjectivity and monotonicity of any alignment function A, when we calculate $Q_{i,j}$, we can assume that $A(j) = i$ and $A(j-1) \in [i, i-1]$. Therefore:

$$
\begin{aligned}
Q_{i,j} &= \max_{A} \sum_{k=1}^{j} \log\left(\mathbb{P}_{\mathcal{N}(\mu_{A(k)},1)}(z_k)\right) \\
&= \max\left(Q_{i-1,j-1}, Q_{i,j-1}\right) + \log\left(\mathbb{P}_{\mathcal{N}(\mu_i,1)}(z_j)\right)
\end{aligned}
\tag{5.1}
$$

This process is shown in Figure 5.2 (Centre).

$Q_{i,j}$ is calculated in an iterative fashion until we reach $Q_{T_{text},T_{wave}}$.

At this point, we work our way backwards to find the optimal alignment A^*. Starting from $A^*(T_{wave}) = T_{text}$, we set $A * (T_{wave} - 1) = T_{text}$ or $A^*(T_{wave} - 1) = T_{text} - 1$ depending on whether $Q_{T_{wave}-1,T_{text}}$ is higher or $Q_{T_{wave}-1,T_{text}-1}$. This step is repeated such that given $A^*(j)$ $\forall j \in [2, T_{wave}]$ we set

$$
A^*(j-1) = argmax_{i \in \{A^*(j), A^*(j)-1\}} Q_{i,j-1} \tag{5.2}
$$

This process is shown in Figure 5.2 (Right).

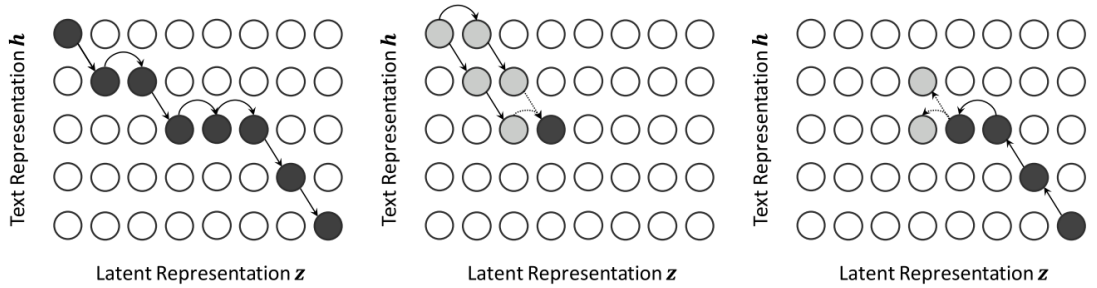

Figure 5.2: Illustration of the MAS algorithm. Left: Example alignment between embedding and latent representation. Centre: Process of calculating the maximum log-likelihood Q. Right: Process of searching for the optimal alignment A^* *Source: Kim et al. (2020)*

5.2.2 Montreal Forced Alignment

An alternative to the MAS algorithm is the Montreal Forced Alignment algorithm ("MFA") which is an open source software produced by McAuliffe et al. (2017). MFA uses the open source automatic speech recognition tool Kaldi (Povey et al., 2011) to first convert audio to a sequence of words and phonemes before aligning the phonemes and words to the spectrogram frames.

We will provide a brief overview of how Kaldi works, before describing MFA. Kaldi tries to find the optimum sequence of words W^* using the mel spectrogram X as follows:

$$\begin{aligned} \mathrm{W}^* &= \arg\max_{\mathrm{W}} P(\mathrm{W} \mid \mathrm{X}) \\ &= \arg\max_{\mathrm{W}} P(\mathrm{X} \mid \mathrm{W}) P(\mathrm{W}) / P(\mathrm{X}) \\ &= \arg\max_{\mathrm{W}} P(\mathrm{X} \mid \mathrm{W}) \quad P(\mathrm{W}) \end{aligned}$$

Where the second equality uses Bayes rule and the last equality uses the fact that $P(X)$ does not depend on W. $P(W)$ is modelled using a hidden Markov model, and $P(X|W)$ is translated to $P(X|Ph)$ where Ph is the sequence of phonemes associated to W using a dictionary such as the CMU dictionary (Weide et al., 2015). $P(X|Ph)$ is then modelled using a Gaussian mixture model and MFA utilises this relationship between X and Ph to align the two.

MFA uses a three step process:

1. The words and phonemes associated with a mel spectrogram is extracted using Kaldi and a phoneme dictionary.

2. Using maximum likelihood and given the fact that the frames of the mel spectrogram are modelled as a Gaussian mixture model dependent on the phonemes, the individual phonemes are aligned to the mel spectrogram frames.

3. The alignment is refined by using a similar process as in Step 2, but using three phonemes in sequence rather than one to utilise the context surrounding the phoneme.

Since each frame in the mel spectrogram can be traced to a fixed range of samples in a waveform, it is easy to convert the alignment to waveforms samples too.

5.2.3 Training the Duration Predictor

Although MAS and MFA are useful for providing alignments between phonemes and mel spectrogram frames or waveform samples, they take a long time to run and are not feasible to be used within the models for inference.

Instead, they are used as estimates of the ground truth durations, and an extra duration predictor learns to map the embeddings extracted from the text encoder to these durations. This means an extra loss component called the duration loss is added to the overall loss function of the diffusion model.

5.2.4 Applying Durations to Embeddings

Once we have the duration for each phoneme, we simply stretch the embedding to ensure that the embedding for each phoneme in the sequence matches the waveform sample(s)

associated to it as shown in Figure 5.3.

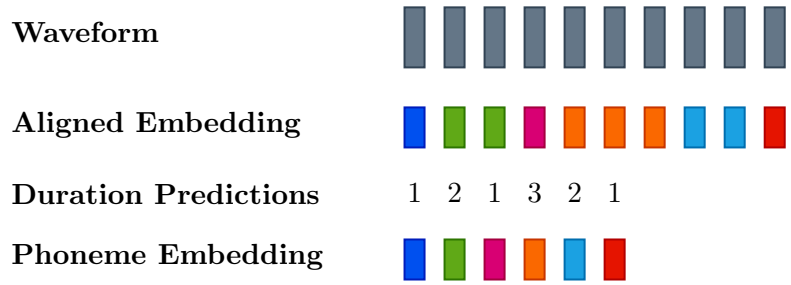

Figure 5.3: Illustration of phoneme embedding being aligned to waveform samples.

The challenge in our particular use case arises from the fact that during training, the inputs are provided by adding noise to $\mathbf{x}_0$ and so the dimensions are independent of the duration predictions. On the other hand, when using a trained model to sample, we wouldn't know the length of the final waveform.

To get around this issue, we apply zero-padding and clipping during training to ensure the dimensions of the aligned embedding match the dimensions of the noisy input when the durations predicted are too short or too long respectively. Our model also outputs the duration predictions separately which allows it to handle scenarios where different examples in a batch may have different sizes. Essentially, zero-padding is used to ensure the dimension matches the largest example and at the end, the predicted durations can be used to remove redundant samples from each example in the batch.

At inference, when we are at the first step of the reverse diffusion process (i.e. when t is equal to the number of transition steps), the initial Gaussian noise is generated by the model based on the duration predictions (since the duration predictor runs before the diffusion model). In subsequent steps, the model simply takes the output from the previous step. This approach allows our model to be robust in handling waveforms of unknown length and being able to cope with mismatching durations during the training process.

As mentioned in the previous section, the duration predictor is trained with a separate loss component called the duration loss. This means that until the duration predictor stabilises the overall model may struggle to learn effectively.

5.3 Bi-directional Dilated Convolution Based Model

In this section we will describe the bi-directional dilated convolutional model and how we adapt it for our use. This model takes in Gaussian noise, and outputs audio with the same dimensions. Our approach is inspired by WaveNet Oord et al., 2016 and DiffWave Kong et al., 2020b since both of these models generate waveforms effectively.

The DiffWave architecture is shown in Figure 5.4. It consists of N residual layers where N is a hyper-parameter that can be tuned. Each residual layer has a channel dimension C, which is also a hyper-parameter.

The first step in the model is to increase the channel dimension of the input by using a 1×1 convolutional layer with C filters. In the following sections, we will explore the components of the DiffWave architecture in detail.

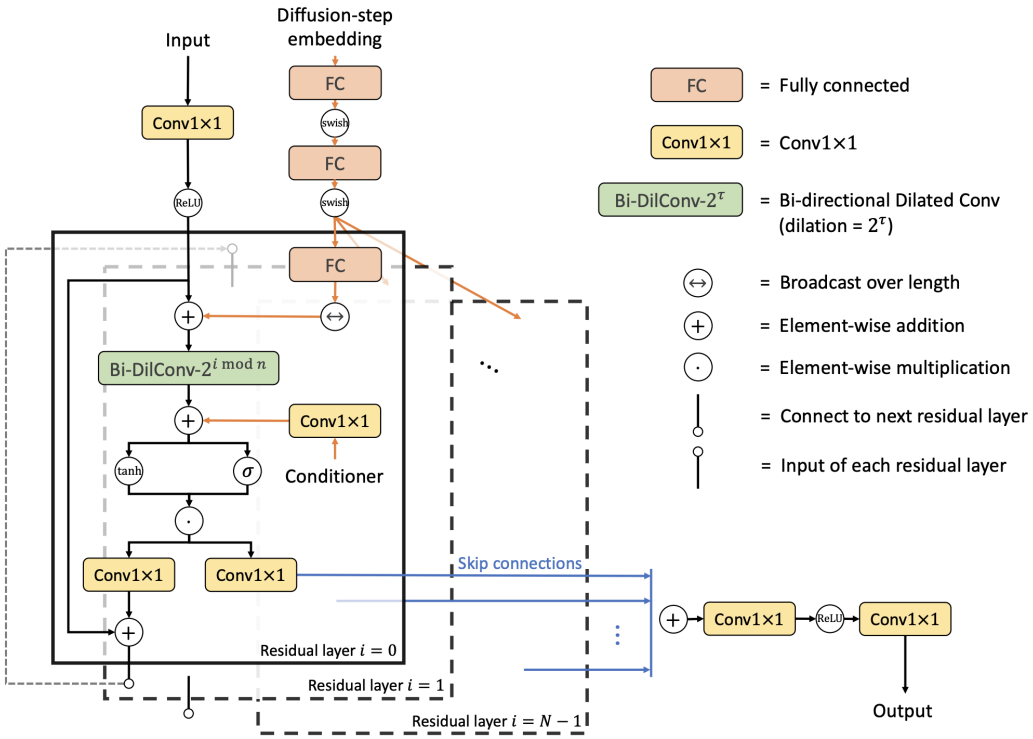

Figure 5.4: Model architecture used in DiffWave. *Source: Kong et al. (2020b)*

5.3.1 Diffusion step embedding

Recall from Section 3.2.3 that the model trained to reverse the diffusion process is conditioned on the time-step embedding. We have adopted the approach suggested by Vaswani et al. (2017) to use sine and cosine functions to create a 128-dimensional embedding as follows:

$$t_{\text{embedding}} = \left[\sin\left(10^{\frac{0\times4}{63}}t\right), \cdots, \sin\left(10^{\frac{63\times4}{63}}t\right), \cos\left(10^{\frac{0\times4}{63}}t\right), \cdots, \cos\left(10^{\frac{63\times4}{63}}t\right)\right] \quad (5.3)$$

We then apply two fully connected layers with swish activation functions followed by a final fully connected layer. The final layer increases the channel dimension of the embedding to match the channel dimension C of the residual layer. The resulting tensor is then added to the input to the residual layer (using broadcasting over the length to match the dimensions).

5.3.2 Conditioner

Some researchers have considered unconditional speech generation for example unconditional WaveNet (Oord et al., 2016). This is where speech is generated without passing any input such as text. In such cases the generated speech resembles human speech, including pauses stresses etc., but does not contain any words that can be understood.

We will not be exploring unconditional speech generation in our research and will always condition the diffusion models as shown in Figure 5.4. Depending on the experiment in question, this conditioner may be a mel spectrogram, or text embedding from the encoder. Other than channel dimensions, the output from the encoder has the same dimensions as the audio since the duration predictor has already stretched the embeddings based on predicted durations.

In the case of mel spectrograms, we configure the properties of the mel spectrograms in such as way that the number of waveform samples is an exact multiple of the number of spectrogram frames. This makes it easy to apply transposed convolutional layers to expand the dimension of the mel spectrogram conditioner appropriately.

Once the length dimension of the conditioner matches the length of the audio waveform, a 1×1 convolutional layer with C filters is applied so that the output can be added to our data before passing through the dilated convolutional layers.

It should be noted that unlike DiffWave, we use the same transposed convolutional layers and 1×1 convolutional layers to process the conditioner for all the residual layers instead of separately processing the conditioner in each layer. This is to simplify the model and reduce the number of parameters.

5.3.3 Residual Layer Architecture

The main component of each residual layer is a Bi-directional dilated convolutional layer. This is similar to the dilated convolutional layer seen in Section 4.1.1 however the layer is bi-directional rather than being causal. The dilation factor is set to $2^{i \mod n}$ where n is another hyper-parameter called the dilation cycle factor. The dilation layer also increases the number of channels to $2 \times C$.

After passing the input through the dilated convolutional layer, the conditioner from the encoder is added after being passed through a 1×1 convolutional layer with $2 \times C$

filters to match the channel dimensions.

This tensor is then split into two chunks of equal size along the channel dimension. The first chunk is passed through a sigmoid activation layer, whilst the second chunk passes through a tanh activation layer acting as the filter. The two chunks are then multiplied element wise.

A final 1×1 convolutional layer is applied to once again increase the channel dimension to $2 \times C$ and the resulting tensor is split into two chunks along the channel dimension. The first chunk acts as the residual element, which is scaled by $\frac{1}{\sqrt{2}}$ whilst the second element forms the skip connection.

The residual element is passed as an input to the next residual layer and the skip connections from all the layers are added together in the end. As mentioned previously, this give the model the ability to observe patterns and periodicity of different lengths and granularities.

5.3.4 Final Projection

Once the skip connections from all the residual layers are added together, the resulting tensor is passed through a 1×1 convolutional layer followed by a ReLu activation layer. A final projection layer reduces the number of channels to 1 forming the final output.

5.3.5 Choosing the Amount of Dilation

As mentioned previously, an important factor to consider is the amount of dilation in each residual layer. Too much dilation could cause the model to be too coarse, and too little dilation would cause the model's receptive field to be too narrow. The way DiffWave was designed with it's skip connections, it is unlikely that the model would be too coarse as the model should learn to place less reliance on the latter residual layers. However it would mean that there are unnecessary layers in the model.

Kong et al. (2020b) show that the total receptive field of a model with the above structure can be calculated as $\sum_i (k_i - 1)d_i + 1$ where k_i and d_i are the kernel size and dilation factor of the ith layer respectively. Kong et al. (2020b) further simplify the hyper parameters by assuming k_i is fixed for all i (they use a kernel size of 3 throughout) and the dilation amount in the ith layer is set to $2^{i \bmod n}$ where n is the dilation cycle length. This reduces the number of hyper parameters down to k, n and the number of layers L.

In our experiments, we decided to keep the kernel size fixed to 3 as well to simplify the convolutions and padding. Kong et al. (2020b) recommend using a receptive field of approximately 6000 samples at 22050Hz. In Table 5.1 we show the proposed hyper-parameters we will use at different resolutions and the resulting receptive fields in terms of number of samples.

Sample Rate	4000Hz	8000Hz	11025Hz	16000Hz	22050Hz
Layers	21	24	30	27	30
Dilation cycle length	8	9	9	10	10
Receptive Field	1083	2171	3081	4347	6139

Table 5.1: Hyper-parameter setting for the dilated convolutional models at different resolutions

5.3.6 Fast Sampling

As detailed in Section 4.3.1, fast sampling can be employed during inference provided a shorter noise schedule. We will therefore use the shorter noise schedule 0.0001, 0.001, 0.01, 0.05, 0.2, 0.5 for our fast sampling experiments.

5.4 Super Resolution Models

As part of the cascaded diffusion model pipelines, we need to train super-resolution models to increase the resolution of the waveforms. These super-resolution models are also diffusion models with the key difference being the way they are conditioned.

In our experiments, we have used two types of super-resolution diffusion models. The first is using U-Nets (see Section 3.3), and the second is using bi-directional dilated convolutions as in Section 5.3. Our rationale for choosing one over the other depended on the particular experiment, and the resolution of audio we were working with.

U-Nets are the models used in the CDMs developed by Ho et al. (2022) and are used in the super resolution model SR3 (Saharia et al., 2022). They have been proven to work extremely well in diffusion models in the image domain due to their ability to condense data to the most important features and reconstruct the data from this condensed version whilst removing noise.

However there are some difficulties in applying U-Nets to audio which we will discuss here. Firstly, it is difficult to estimate the receptive field of a U-Net model and as discussed earlier (Section 5.3.5) this is an important aspect of audio models. Secondly, U-Net based models are much larger than the bi-directional dilation based models discussed in Section 5.3 and the large number of parameters slows down training and inference.

Lastly, in Section 5.3 we described how we re-used the same network to project our conditioning mel spectrograms to reduce the number of parameters. In U-Nets, each of the residual blocks should be given the context, i.e. the conditioning mel spectrogram or speech encoding. Since the residual blocks have different dimensions, this means there is a need to have a different projection network for each block. This further increases the number of parameters and complicates the network further.

5.5 Dataset

The primary dataset we will use is the LJ Speech dataset (Ito and Johnson, 2017) which is a public data set available at https://keithito.com/LJ-Speech-Dataset/.

This dataset contains c.13k recordings which are up to 10 seconds long of the same speaker, Linda Johnson, reading from a selection of 7 books. Each recording is a single-channel 16-bit PCM WAV file with a sample rate of 22050Hz.

The dataset is commonly used to test text to speech models due to the fact that it contains over 24 hours of recordings in total and the accompanying text is provided. For example, most of the models described in Section 4 are trained on the LJ Speech Dataset including the diffusion models DiffWave (Kong et al., 2020b) and Grad-TTS (Popov et al., 2021).

The LJ Speech dataset contains two versions of text for each audio: the actual text from the book; and a normalised version where for example numbers are converted to their alphabetical forms. We have used the latter version throughout as the alphabetisation aids us in obtaining phoneme representations.

5.6 Evaluation Metrics

In this section we will describe some of the evaluation metrics we have used, the reason we have used them as well as the limitations of these metrics. These metrics will help us assess the quality of generated speech, and ultimately help us understand whether our approach is successful or not.

5.6.1 Mean Opinion Score

When it comes to speech generation, and indeed sound generation in general, the best way to evaluate the output is through human ears. Whilst there are some metrics that can give an indication as to the quality of generated audio, humans are naturally able to perceive imperfections in tone, pronunciation, stresses and emphasis. For this reason, all of the models in Section 4 rely on human listeners to evaluate the audio.

This is usually done by using crowd sourcing on platforms such as Amazon Mechanical Turk ("MTurk"). People that accept the jobs (called "workers" on MTurk) are given audio samples generated from a number of different models including the ground truth audio and asked to evaluate them with a score between 1 and 5 where 5 represents natural audio and 1 represents generated audio. These scores are averaged to provide a mean opinion score ("MOS") for each model.

Clearly, this evaluation can be subjective and so it is important to use a large number of workers and provide them with a sufficiently large sample of audio from each model.

For example, Popov et al. (2021) use 10 workers and found a MOS of 4.53 ± 0.06 for the ground truth LJSpeech dataset (Ito and Johnson, 2017) whereas Ren et al. (2019) observed a MOS of 4.41 ± 0.08 for the same dataset using 20 workers.

In addition, it is important to select workers with a higher proficiency in the language of generation, in our case English, so that they are better able to identify pronunciation issues.

All of these factors increase the cost of using MTurk making it infeasible for an MSc project. We therefore used a survey sent to family and friends as a proxy and asked people to evaluate audio from different experiments. Everyone completing the survey had a minimum of an English taught Bachelor's degree and so were sufficiently qualified to rate the audio clips.

We chose 5 samples for each model/configuration including the ground truth audio and received 41 responses. The order of the audio was randomised and all labels were removed. Typically, when getting MOS scores, assessors are sent at least 40 samples per model, however as we were asking people to volunteer their time, this would not have been feasible. This inevitably caused an increase in the variance of our scores.

5.6.2 Fréchet Audio Distance

Fréchet audio distance ("FAD") is a metric proposed by Kilgour et al. (2019) to evaluate the quality of generated audio. According to the authors, this method of assessing audio correlated better to human opinions than previous methods such as cosince similarity when it came to detecting a range of distortions in audio.

The overall FAD methodology is described in Figure 5.5 for the use case of evaluating the quality of music. Unlike other metrics, FAD doesn't assess individual audio, but rather compares the space of audio generated by a particular model and to the space of audio of the ground truth. The idea is to use a pre-trained audio classification model to generate embeddings of the audio to be able to compare these spaces.

Kilgour et al. (2019) use the final layer of the VGGish audio classification model (Hershey et al., 2017) to extract embeddings and we will do the same. The VGGish model was trained on audio from 70m Youtube videos to classify them into around 30k categories.

We start by calculating the multivariate Gaussian distributions $\mathcal{N}_e\left(\mu_e, \Sigma_e\right)$ and $\mathcal{N}_r\left(\mu_r, \Sigma_r\right)$ for the embeddings of the evaluation set and reference set respectively using maximum likelihood estimation.

According to Dowson and Landau (1982), the Fréchet distance between two multivariate Gaussian distributions is calculated as follows:

$$\mathbf{F}\left(\mathcal{N}_b, \mathcal{N}_e\right) = \|\mu_b - \mu_e\|^2 + \mathrm{tr}\left(\Sigma_b + \Sigma_e - 2\sqrt{\Sigma_b \Sigma_e}\right) \tag{5.4}$$

The VGGish model is very restricting in terms of the input (it takes in audio of length 1 seconds at a sample rate of 16kHz converted to mel spectrograms) and Kilgour et al. (2019) recommend using 0.5s overlapping windows to extract the 1 second audio from larger samples.

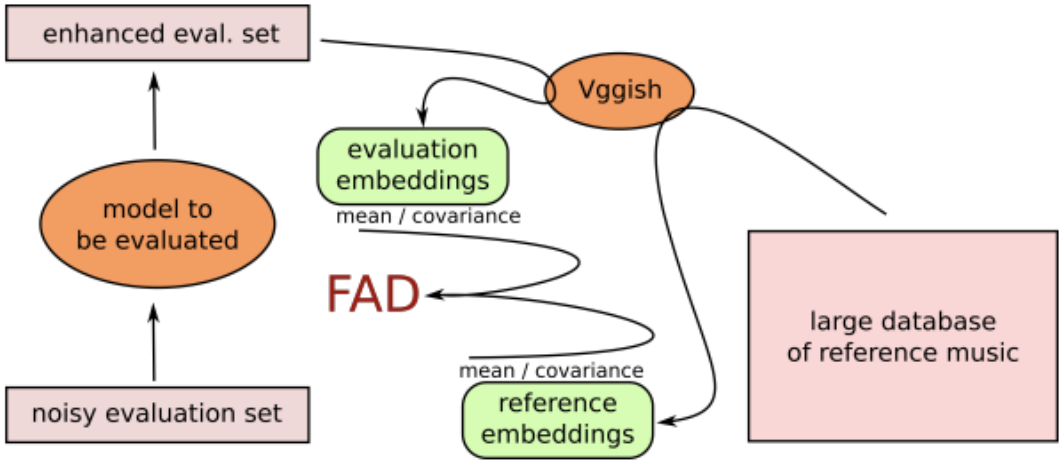

Figure 5.5: Overview of FAD for music quality evaluation. Using a pre-trained audio classification model, VGGish, embeddings are extracted for both generated audio as well as the ground truth audio before the two are compared. *Source: Kilgour et al. (2019)*

5.6.3 Other Metrics

As mentioned in the previous section, the FAD score was reported by Kilgour et al. (2019) to be the closely correlated to human opinion. However, we will also look at additional metrics at individual audio level for completeness. These are described in the following sections.

Cosine Distance

One way of determining the similarity between a generated waveform and the ground truth waveform is through cosine similarity:

$$\text{cosine similarity}\,(w_1, w_2) = \frac{w_1 \cdot w_2}{\|w_1\|\,\|w_2\|} \tag{5.5}$$

We utilise the cosine similarity metric described in Equation 5.5, but instead set the cosine distance equal to 1 - cosine similarity. This means our metric takes values between 0 and 2, and similar to FAD, lower values are preferred.

Unlike images, where root mean square error ("RMSE") can be used to compare the error between the generated image and ground truth, comparing waveforms directly is a difficult task. For example calculating the cosine distances on audio directly as slight distortions or lags may cause the cosine distances to explode. Moreover, where we are predicting durations, the lengths of our predicted audio may not even match the length of the ground truth audio. To solve this issue, we calculate cosine distances on the embeddings extracted from the VGGish model similar to FAD.

Source to Distortion Ratio

We considered using metrics to measure the level of noise in the generated audio compared to the ground truth. For example, Vincent et al. (2006) introduced a sound to distortion ratio ("SDR") that is particularly effective at assessing signal separation.

However we decided against using such metrics for the same reason we did not apply cosine distances directly to the waveforms; SDR works well in signal separation because there are no alignment issues but that isn't necessarily the case for generated audio.

5.7 Training

All of our models are trained on the entire LJSpeech dataset except for 523 audio clips. According to Kilgour et al. (2019), a sample of at least 300 audio clips with an average length of 5 seconds should be used to ensure the FAD scores are stable. Therefore our hold out set of 523 clips is sufficient for this evaluation.

One of the difficulties in training diffusion models is there is no validation score that can be used to indicate when to stop training the model. The model needs to learn how to make predictions at each of the time steps, and so validation score would need to be evaluated on all of the time steps on the validation set. Moreover, there is no accuracy metric that can be calculated for audio generation, and the evaluation metrics above only give us an indication.

We therefore use the approach used by Popov et al. (2021) and Kong et al. (2020b) and train the model on a batch size of 16 for 1m steps with a learning rate of 2e-4 and the Adam optimiser. In addition, we save the models intermittently and listen to the audios reproduced by these intermediate models to see if they perform better than the 1m steps

model. However in all of the models we trained, we found that the model after 1m steps generally sounded better.

6 Experiments

In this section we will describe the experiments we have conducted as part of our research, and evaluate the results, suggesting any areas that could be explored in the future.

We first recall that our research goal is to investigate the use of CDMs in speech synthesis. In particular we want to investigate whether CDMs can be used to generate audio effectively from low resolution mel spectrograms; and whether CDMs can improve the accuracy of generated speech.

To test the first point, we will use mel spectrograms relating to 8kHz audio and use these to ultimately generate speech audio with a resolution of 16kHz in Experiment II below. Generating high resolution audio from low resolution mel spectrograms has two benefits: the approach can be used to generate high fidelity speech using existing models generating mel-spectrograms; or reduce the size of the base model that generates mel-spectrograms from text which tends to be the largest model and thus reduce training times.

To test the second point, in Experiment II we will create an end to end text to speech pipeline using the CDM approach and see if this provides any accuracy benefits over existing models. Lastly, in Experiment III we will investigate whether a CDM can be used to generate speech directly from text without the need to generate mel spectrograms first.

Note that sample outputs from all the experiments can be found at https://on.soundcloud.com/xxEd6.

6.1 Experiment I: Cascaded Diffusion Models Involving Low Resolution Mel Spectrograms

In this experiment, we wanted to test whether we could generate low resolution speech from low resolution mel spectrograms, and then apply a super resolution model to output high resolution audio. In a pipeline of cascading models comprising a text to speech system, the first model, usually the model that generates the mel spectrogram from text, is the most crucial. This is because this model learns the high level features such as the tone of voice, the breaks and stretches etc. and these are the attributes that our ears perceive first.

Naturally, these models need to be tuned carefully which takes a considerable amount of time. By reducing the resolution of these models and reducing their parameters, it becomes easier to train them and tune them to different datasets/speakers. The question we want to answer in this experiment is whether the final output using this approach can compare to using a higher resolution base model.

6.1.1 Data Pre-processing

The only data pre-processing we conduct is to resample the audio to a resolution of 8kHz and generate the mel-spectrograms from these lower resolution audio clips. Typical values for mel-spectrograms use a hop size of 256 samples, and a window size of 1024 samples for 22050Hz audio. Due to the lower resolution that we are working with, and since we don't want to violate the periodicity assumptions of Theorem 2.2.1, we halve these values.

In addition, once the base model is trained, we use cubic interpolation to generate the inputs for the super resolution model as discussed in Section 3.4.

6.1.2 Approach

To conduct this experiment, we train a pipeline of two models that will take as input mel spectrograms corresponding to audio with a sample rate of 8kHz to generate audio with a frequency of 16kHz.

The first model generates waveforms with a sample rate of 8kHz. This model is able to learn the high level details of the speech and the output is intelligible speech audio even if it is noisy. The second model then acts as the super-resolution model in the pipeline.

Base Model

The base model, i.e. the model that generates the 8kHz waveform is based on the architecture described in Section 5.3. We use a dilation cycle length of 9 with 24 residual layers. We set the residual channel size to 64 due to the lower resolution.

Super Resolution Model

For the super resolution model, we adopt the U-Net model described in Section 3.3.1. In order to reduce the size of the model and the training and inference times, we use four down-sampling blocks and four up-sampling blocks. These blocks consists of 1-dimensional convolutional layers instead of 2-dimensional layers seen in typical U-Nets.

Similar to the rationale used by Donahue et al. (2018) when modifying the DCGANs to be used for audio, we increased the up-sampling rates of the residual blocks to account

for the large number of data points in audio data and to help increase the receptive field of the U-Net.

Note that we do not condition our U-Net on the low resolution mel spectrograms. This is because the waveforms generated by the base model are already used as an input to the super resolution model, and these waveforms should contain more information than the corresponding mel spectrograms.

6.1.3 Results

The audio produced by our model was very intelligible, and the speech resembled the original speaker's prosody very well. However we noticed the presence of white noise in the audio which seems to suggest that our model may not be detecting and removing all of the noise appropriately.

The results of our evaluation are shown in Table 6.1 as "CDM (Low Res)". We have also used a pre-trained DiffWave (Kong et al., 2020b) model for comparison. We noted that the FAD and cosine distances of our CDM model were significantly larger which coincides with the white noise that could be heard in the audio. The MOS score of our model is also worse than that of the DiffWave model, but not to the extent of the FAD and cosine distance scores which may suggest that these scores are overly sensitive.

To investigate the cause of the performance issues, we first needed to identify which stage of the CDM is under performing. To do this, we re-evaluated our results for the base model in the pipeline to produce waveforms with a resolution of 8kHz. We then down sampled the ground truth audio and compared the two. The results are shown in Table 6.1 under "CDM (Low Res Base Only)". We can see that at the 8kHz resolution our model performs similarly to the DiffWave model. This suggests that the performance issues are caused by our super resolution model.

Model	FAD	Cosine distance	MOS
Ground Truth	-	-	4.22 ± 0.11
DiffWave	1.88	0.074	3.96 ± 0.11
CDM (Low Res)	9.18	0.203	3.16 ± 0.12
CDM (Low Res Base Only)	1.24	0.075	-

Table 6.1: Results of the low resolution CDM model compared to the DiffWave model

There are a number of possible reasons why our super resolution model may not be performing as well. One reason may be due to the fact that we restricted the size of the model to speed up training which could mean the model performance is adversely impacted. Another reason may not be the size of the model per say, but the way the size was reduced (i.e. by including one less down sampling and up sampling blocks compared

to SR3 Saharia et al. (2022)) which results in a smaller receptive field of the model.

Indeed it may be the case that U-Net architectures are simply not sufficient for the task of audio super-resolution as they do not offer the large receptive fields that come with other types of architectures such as dilated convolution based models. Recent research by Lee and Han (2021) indicates that dilated convolution models show promising results when up-sampling audio from 16kHz to up to 44.1kHz and future studies should investigate whether their approach works with lower resolution audio at 8kHz.

To illustrate the performance of the base model in our CDM pipeline, we show how the waveforms predicted by the model change during the reverse diffusion process in Figure 6.1. From this diagram we notice that the waveform is very noisy even after 30 out of the 50 steps and only starts to resemble the ground truth waveform towards the end.

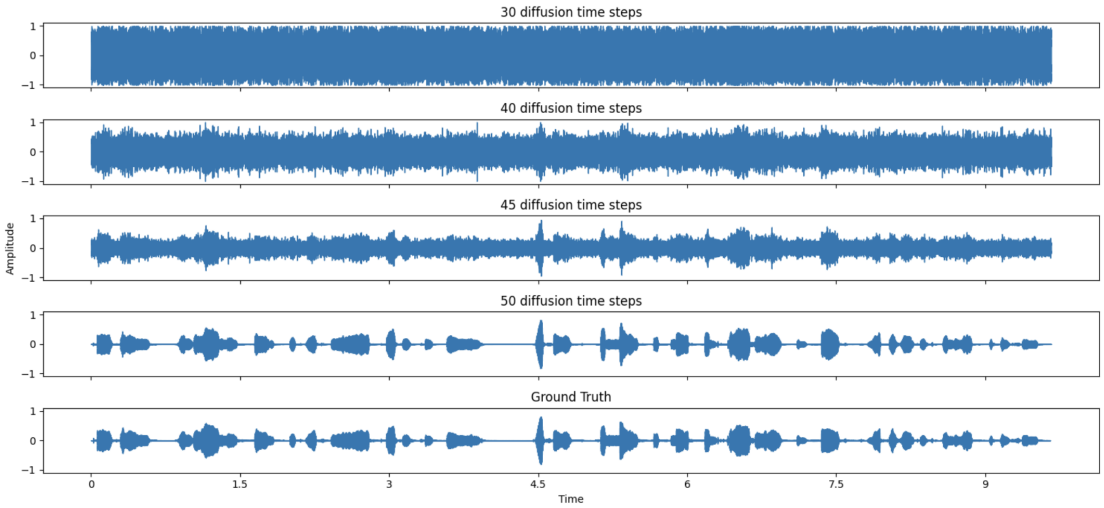

Figure 6.1: Illustration of the reverse diffusion process for generating a sample waveform using the base model of Experiment I.

6.2 Experiment II: Text to Speech (End to End) Using Cascaded Diffusion Models

For our second experiment, we follow a similar approach to Experiment I (Section 6.1) however we don't restrict our selves in terms of the resolution of the mel spectrograms. In other words, our starting point is a mel spectrogram corresponding to audio with a resolution of 22050Hz. The output of our CDM pipeline in this experiment is audio with a resolution of 22050Hz.

The reason for producing higher resolution audio in this experiment is so that we can use a pre-trained model such as Grad-TTS (Popov et al., 2021) to produce the mel

spectrograms rather than obtaining these directly from the audio. This allows us to test the end to end text to speech pipeline.

6.2.1 Data Pre-processing

We follow a similar approach to Experiment I for our data processing except this time we generate our mel spectrograms from 22050Hz audio and therefore use the more standard window size of 1024 samples and hop size of 256 samples.

6.2.2 Approach

Our pipeline consists of three models: a pre-trained Grad-TTS model (Popov et al., 2021) which generates 22050Hz mel spectrograms; a dilated convolution based model that generates 11025Hz waveforms from the mel spectrograms (see Section 5.3); and finally a super-resolution model that returns 22050Hz waveforms. In the following sections we will cover each of the models.

Base Model

The base model we use is a pre-trained Grad-TTS model (Popov et al., 2021) which can be found on https://grad-tts.github.io/. Apart from the fact that the authors boast impressive results from Grad-TTS, our reasoning for using this model is due to the fact that it is the only mel spectrogram generating diffusion model to our knowledge that has a pre-trained model created by the authors of the paper themselves.

Since this is the base model, there was not much benefit to training our own model to generate the mel spectrograms as our model would not differ greatly.

We could have used a non-diffusion based model but the advantage of using a diffusion model is that inputs to the models that follow in the pipeline can be augmented (See section 3.4) which Ho et al. (2022) suggest can improve the pipeline as a whole.

Grad-TTS offers the ability to scale the durations of the predicted audio by multiplying them by a user-defined scalar. We ran initial tests and found that setting the scale factor equal to one produced audio that was unnaturally stretched in certain parts. After sampling a handful of audio with other listeners, we decided to use a scale factor of 0.8.

Grad-TTS can also be used to tune the number of diffusion steps, and we experiment with using 100 steps vs 1000 steps as recommended by the authors.

Waveform Generating Model

For the first model that generates waveforms, we used the same approach as in Experiment I (Section 6.1) but re-trained the model based on the new resolutions. We used a cycle length of 9 with 29 residual layers for this model. Ideally we would have used a residual channel size of 128 as recommended by Kong et al. (2020b) however we restricted this size to 64 to restrict the number of parameters.

Note that in Experiment I, the goal was to test waveform generation from mel spectrograms whereas in this Experiment our starting point is the input text. Therefore in an ideal scenario, we would have trained this models on the outputs of the previous model in the pipeline.

In practice, this is difficult to implement due to the variance in durations predicted by the first model. The durations predicted by the first model, and therefore the number of frames in the mel spectrogram outputs do not align perfectly to the number of samples in the waveforms being predicted by this model. We found that the total duration of the equivalent audio could differ by up to half a second (c.5500 samples).

One simple solution could be to apply zero-padding or cropping. Alternatively, a more elaborate solution would have been to tweak the scale factor for each sample in our training set to align it to the target waveforms of the next model and then apply a minimal level of cropping or zero-padding.

However, for computational reasons we take short portions of the audio for training which are around 1 second long each. Therefore even if we force the overall shapes of the inputs and outputs to align, when we take the 1 second samples, there could be a big misalignment between the inputs and outputs. For this reason, we have decided to train this model on mel spectrograms generated directly from the target audio which is the standard approach used to train most vocoders (Kong et al., 2020b), (Kong et al., 2020a), (Donahue et al., 2018). In Experiment III (Section 6.3) we discuss ways in which these misalignment issues can be managed in future experiments.

Super Resolution Model

After Experiment I, we have decided to use a dilated convolutional approach for our super resolution model. Given that in this experiment we are working with higher resolution audio, having a large receptive field would be even more important and a U-Net based approach may not suffice unless we make the model very deep.

We decided to use a cycle length of 10 with 30 layers which is in line with the parameters chosen by Kong et al. (2020b) for 22050Hz audio generation. Again, we restrict the channel size to 64 which may restrict performance. Unlike the super-resolution model used in Experiment I, this model utilises the mel spectrogram in addition to the up-sampled audio from the previous model in the pipeline. We use the approach recommended by

Ho et al. (2022) and Saharia et al. (2022) and concatenate the up-sampled audio with the noisy inputs along the channel dimension.

This concatenated data passes through the initial convolutional layers to increase the number of channels to match the channels in the residual layers, where the projected mel spectrograms are added as described in Section 5.3. Although this approach for concatenating the data worked well for images, we would recommend trying to use more sophisticated approaches such as having a separate projection layer for the up-sampled audio and concatenating this to the projected mel spectrograms along the channel dimension to see if this results in any improvement in model performance.

6.2.3 Results

We show the model trained under this experiment evaluated under our evaluation metrics in Table 6.2. In the table we have shown different version of the same pipeline as follows:

1. CDM is our end to end model as described above which has been evaluated on it's ability to take text from the test set and convert this to speech.

2. CDM Vocoder is our model without the Grad-TTS component. This is to compare how our model compares to waveform generation using perfectly constructed mel spectrograms from audio rather than outputs from Grad-TTS.

3. CDM Fast is the same as the CDM model but uses fast sampling throughout.

4. CDM Vocoder Fast is the same as CDM Vocoder but uses fast sampling throughout.

5. CDM Semi-Fast 1 is the same as the CDM model, but uses fast sampling fast sampling in the dilated convolution models only (the Grad-TTS model still uses 1000 timesteps).

6. CDM Semi-Fast 2 is the same as the CDM model, but uses fast sampling for the Grad-TTS model (the dilated convolution models still use normal sampling).

We have also included the DiffWave vocoder, and Grad-TTS and HiFi-Gan models for comparison. The table also shows inference times, which was calculated as the average inference time on 100 runs on a 9 second long audio.

From the results in Table 6.2 we see that Grad-TTS with the HiFi-GAN vocoder has the lowest FAD but DiffWave has a lower cosine distance. Generally, our models have higher FAD and cosine distance values than Grad-TTS with the HiFi-GAN vocoder or DiffWave. However our models tend to have slightly higher MOS values. However as mentioned earlier, due to the limited sample sizes the variances are quite high as can be seen by the 95% confidence intervals shown in Table 6.2.

Model	FAD	Cosine distance	MOS	Inference time (s)
Ground Truth	-	-	4.22 ± 0.11	-
Vocoders				
DiffWave	1.88	0.074	3.96 ± 0.11	5.5
CDM Vocoder	2.99	0.098	4.08 ± 0.13	32.2
CDM Vocoder Fast	3.16	0.103	3.78 ± 0.12	1.0
End to End TTS				
Grad-TTS + HiFi-GAN	0.58	0.086	3.48 ± 0.13	21.3
CDM	2.91	0.137	3.69 ± 0.13	53.5
CDM Fast	3.01	0.138	3.59 ± 0.12	3.1
CDM Semi-Fast 1	3.00	0.139	3.56 ± 0.13	22.3
CDM Semi-Fast 2	2.91	0.137	3.57 ± 0.14	34.3

Table 6.2: Results of the CDM compared to DiffWave and Grad-TTS

After examining the audio produced by the models, it appears that the CDM approach may produce slightly better speech, however there is a presence of white noise (albeit lower than the low resolution CDM model from Experiment I). This may suggest that the chaining of diffusion models is causing noise to build up resulting in adverse FAD and cosine distance values. It would therefore be worthwhile exploring conditional augmentation as recommended by Ho et al. (2022) as this may better equip the models to deal with and correct for noisy inputs.

We also noted from the results that generally, the vocoders performed better than the end to end models in terms of MOS. This was to be expected since the vocoders use perfectly constructed mel spectrograms rather than estimates from another model in the pipeline. We also noted that the fast inference did not result in a huge drop in MOS. Notably, in the text to speech models, even the fast inference CDMs outperform Grad-TTS + HiFi-GAN in terms of MOS.

This seems to suggest that the DiffWave style vocoder performs better in terms of human perception than HiFi-GAN, even with fast sampling. However, one issue with the CDM approach appears to be the fact that multiple diffusion models running in sequence cause the inference times to add up. This may not be an issue with image generation, but for speech synthesis latency can be an important factor. Having said that, as mentioned our fast CDM still outperforms Grad-TTS + HiFi-GAN and has a much lower inference time. Thus with the help of fast sampling, and at the slight expense of quality, CDMs are still viable in this setting.

6.3 Experiment III: Direct Waveform Generation Using Cascaded Diffusion Models

In this chapter, we conducted an additional experiment and created a CDM for the use case of generating speech from text directly. This is a task that most models have struggled with, and the only model that has been able to do this successfully has been WaveNet (Oord et al., 2016). However WaveNet required a significant amount of feature engineering which can be difficult to implement in practice.

We therefore wanted to test whether a CDM would be able to perform the task better by solving the problem in a coarse manner first and refining the results in subsequent higher resolution models.

The overall architecture is summarised in Figure 6.2 and in the following sections, we will go over each of the components in detail.

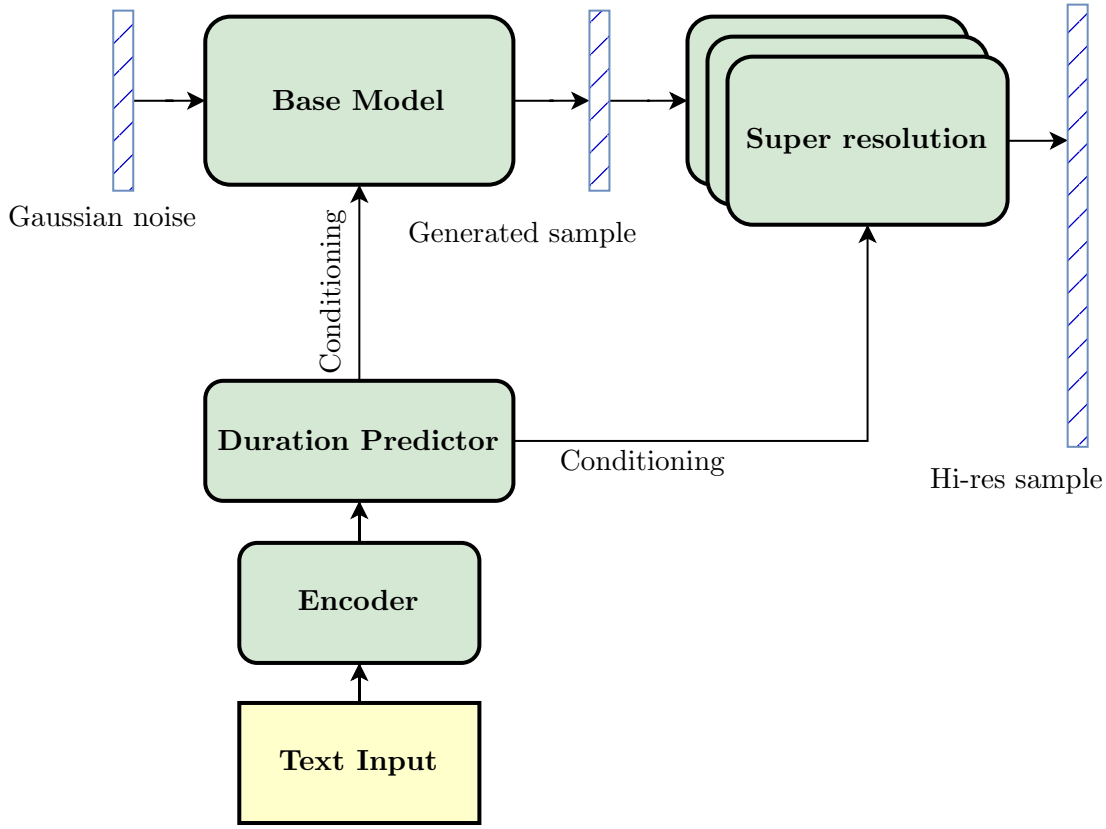

Figure 6.2: Proposed model architecture for speech generation using CDMs

6.3.1 Base Model

Our base model was designed to produce waveforms with a sample rate of 4000Hz. We experimented with the use of even lower resolutions but found that the audio was not intelligible beyond this point and therefore would have been difficult to evaluate. We use the text encoder described in Section 5.1 which creates d-dimensional embeddings which are then aligned to the waveform samples by the duration predictor.

Duration Predictor Using MAS

We initially used the MAS algorithm to train our duration predictor as recommended by Kim et al. (2020) and Kong et al. (2020b). However we found that this approach did not fit in with our particular use case.

Firstly, as detailed in Section 5.2.1, the MAS algorithm works by aligning mean values using maximum likelihood estimation and our encoder's objective isn't necessarily to produce mean values as is the case in Grad-TTS. Therefore we added a series of two convolutional layers with ReLU activations which reduced the channel dimension to $\frac{d}{2}$ and then to 1. An additional element was added to the loss function to encourage the model to get this output as close to the mean values as possible. We added a gradient break between the encoder and these additional layers to ensure that the encoder was not impacted by this additional goal.

Based on the mean values predicted by these additional layers, MAS was used to find the target alignment during training. We then added two convolutional layers with ReLU activations to learn the durations being estimated by the MAS algorithm. A further loss element was added which measured how far the predicted durations were from those obtained from the MAS algorithm.

This workaround added unnecessary complexity to the model, making it difficult to debug. In addition, we found that the duration predictions failed to stabilise. This is likely caused by two issues. Firstly the encoder has a different objective which is to enable the base model to perform diffusion as per Algorithm 1, but relies on the duration predictions. Therefore the encoder can't stabilise until the duration predictions stabilise and the duration predictions which relies on the encoder can't stabilise until the encoder itself stabilises. Popov et al. (2021) reported that their model suffered from the same issue and therefore took much longer time to train before stabilising.

The second reason is that waveforms have a high variance even at a resolution of 4kHz and the mean values tend to remain close to 0. In this case, the assumption made in the MAS algorithm to assume standard deviation is 1 is probably too restricting. Thus using the MAS algorithm on waveforms is unlikely to produce reliable results. We therefore opted to use a different approach as described in the following section.

Duration Predictor Using MFA

As described in Section 5.2.2, the open source MFA tool can be used to extract duration predictions by aligning phonemes to the waveforms. The main advantage of this approach is that it can be done independently from our diffusion model and so the ground truth durations can be passed as an input during training.

In fact, we took this approach one step further to speed up training and assumed that the durations would be provided as an input to the diffusion model which meant the diffusion model had one objective and one loss function as per Algorithm 1.

Since the MFA tool is slow to run and difficult to incorporate into a pipeline as it is a command line tool, we still trained a duration predictor separately. This duration predictor took the phonemes as inputs and predicted the durations for each. For this duration predictor, we used a pre-net with a convolutional layer and ReLU activation. This was then followed by three residual layers with dilated convolutional layers with dilations values 2, 4 and 8. This can be thought of as a smaller version of the model described in Section 5.3.

We found that this duration predictor stabilised very quickly and produced results with an error rate of 34 samples per average. Even at a sample rate of 4000Hz, this level of prediction accuracy is sufficient and would not be noticeable by human ears.

The advantage of training the duration predictor separately in the context of CDMs is that for the diffusion models, we could use the durations from the MFA algorithm which align perfectly to the total number of samples in the waveform. This made it easier to chain together our diffusion models as the sizes of the audio were fixed rather than being dependent on a trained duration.

Once we have the embeddings from the text encoder, and these are aligned using the durations which we take as inputs (from the MFA tool during training and from our duration predictor during inference), the next step is to process these in line with Algorithm 1 as described in the next section.

Model Core

The "core" of the model is the part that takes the embeddings to perform the diffusion process. We used the dilated convolutional model architecture described in Section 5.3 which takes a noisy input with the same dimension as the target waveform.

This model is conditioned on the output of the text encoder. Recall from Section 5.3 that the the conditioner is added to each residual block and so we set $d = 2 \times C$ to ensure the channel dimensions match and there is no need to apply additional layers.

Although Kong et al. (2020b) recommend setting C to 128, we used a value of 64 to reduce the number of parameters and reduce the training timescales.

6.3.2 Super Resolution

Although the architecture described in Figure 6.2 allows the use of multiple super resolution models (for example to increase the resolution to 8kHz and then to 16kHz), we simplified the approach to use one super resolution model and increase the resolution directly to 16kHz.

The architecture of this model is the same as that used in the super resolution model from Experiment II (Section 6.2. The only difference is that instead of being conditioned on mel spectrograms, we output the embeddings from the base model to be re-used in this model. These embeddings pass through a transposed convolutional layer to increase the dimension by a factor of four to align them to the new resolution of 16kHz.

6.3.3 Results

We used audio perception to evaluate the outputs of our model and found that in the majority of cases the speech was not intelligible. Some words could be understood, but it was difficult to understand the entire sentence beyond this. We therefore did not conduct FAD scores or include this audio in our MOS survey. The audio appears to have the same prosody as the ground truth and so the model has learnt the prosody well, but it almost sounds as if the speech is in another, unknown language.

We found that the main issue was down to the base model and even at 4kHz, the model was unable to reproduce the waveforms well. The super resolution model could not correct for the errors of the base model as expected. Indeed in the cascaded model approach used by Ho et al. (2022), the models at each resolution could be thought of as standalone models which produced competitive results at those resolutions. This means that for our CDM to perform well, it is crucial for the base model to also perform well.

Our results show that even at lower resolutions, the task of generating waveforms directly is not straightforward or easy. The models need to be guided by lower detail representations of the waveforms for example in the form of mel spectrograms or in the form of extensive feature engineering (Oord et al., 2016).

Although it is possible that limitations in model sizes could have influenced our results, judging by the difference in results achieved in this experiment as compared to our previous experiments, it is unlikely that simply using more parameters is going to fix the issue.

7 Conclusion and Further Research

7.1 Conclusion

In this thesis, we have shown that CDMs are a viable option for speech generation and have created the first end to end diffusion based text to speech pipeline. Our results show that whilst some further research and refining may be required, the use of our end to end CDM which incorporates Grad-TTS Popov et al. (2021) had the highest MOS score and therefore has the potential to be a leading approach in terms of speech quality. Moreover, we showed that our CDM can be tuned by the user to find the right balance between inference times and quality.

We did note that inference times can increase due to the fact that multiple diffusion models need to be run in sequence. However our experiments still showed worthwhile results when using fast sampling at the expense of quality.

With further research into improving the super resolution models, we have shown that our lower resolution CDM approach may be able to generate high resolution audio with acceptable quality levels. This could enable the use of smaller mel spectrogram generating models that can be trained more easily and thus tuned more freely before the super resolution models add the finishing touches.

Lastly, while our attempts to use CDMs to directly generate waveforms were unsuccessful, we stumbled on a novel way to generate phoneme durations separately to allow easier integration and alignment between models in a CDM pipeline during training.

7.2 Further Research

Whilst we have proven the viability of CDMs for speech generation, there is a lot of further research that can be performed to refine our models or to explore different approaches. We will list some of these ideas in this section.

7.2.1 Conditional Augmentation

Ho et al. (2022) recommend the use of conditional augmentation, and experimenting with different levels of noise injection in between models in a CDM pipeline. Due to the amount of time it takes to train a single diffusion model, it was not feasible for us to

experiment with different levels of conditional augmentation. However Ho et al. (2022) found this to be a crucial part of the CDM pipeline and research should be conducted to see if this results in any improvements in the speech generation task.

7.2.2 Mel Spectrogram Super Resolution

For this thesis, we have focused on applying super resolution models to waveforms as part of the CDM pipeline. However some researchers have taken the approach of applying super resolution models to the mel spectrograms instead Sheng et al. (2019) and this alternative approach could be explored to see if it results in a difference in results.

7.2.3 Experimentation with Larger Models

As noted in our experiments, we restricted the size of our models to reduce training times, however given the nature of the task this is likely to have adversely affected the performance of our models. It would therefore be useful to re run the experiments with a larger set of parameters to see if the unrestrained models improve the performance.

We also made other simplifications to our models such as fixing the kernels in the residual layers of the bi-directional dilated convolutional model. Exploring different kernel combinations as suggested by Kong et al. (2020a) may further improve the model performance.

7.2.4 Generalisation of Approach

In this thesis, we have conducted our experiments on a single speaker dataset, and further research should be conducted to assess the viability of CDMs in adaptive text to speech, where the model learns to predict speech according to particular speakers.

Moreover, whilst we have kept feature engineering to a minimum, we still rely on phoneme dictionaries. Whilst these are not difficult to use, and do not require expert domain knowledge, they can be restrictive in terms of the languages in which speech could be generated as such dictionaries are not available in every language. Therefore investigating how the models can reduce reliance on such dictionaries would enable text to speech models such as our CDMs to be more accessible.

Bibliography

Chris Donahue, Julian McAuley, and Miller Puckette. Adversarial audio synthesis. *arXiv preprint arXiv:1802.04208*, 2018.

DC Dowson and BV666017 Landau. The fréchet distance between multivariate normal distributions. *Journal of multivariate analysis*, 12(3):450–455, 1982.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Shawn Hershey, Sourish Chaudhuri, Daniel PW Ellis, Jort F Gemmeke, Aren Jansen, R Channing Moore, Manoj Plakal, Devin Platt, Rif A Saurous, Bryan Seybold, et al. Cnn architectures for large-scale audio classification. In *2017 ieee international conference on acoustics, speech and signal processing (icassp)*, pages 131–135. IEEE, 2017.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *The Journal of Machine Learning Research*, 23(1):2249–2281, 2022.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Keith Ito and Linda Johnson. The lj speech dataset. `https://keithito.com/LJ-Speech-Dataset/`, 2017.

Kevin Kilgour, Mauricio Zuluaga, Dominik Roblek, and Matthew Sharifi. Fréchet audio distance: A reference-free metric for evaluating music enhancement algorithms. In *INTERSPEECH*, pages 2350–2354, 2019.

Jaehyeon Kim, Sungwon Kim, Jungil Kong, and Sungroh Yoon. Glow-tts: A generative flow for text-to-speech via monotonic alignment search. *Advances in Neural Information Processing Systems*, 33:8067–8077, 2020.

Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. *Advances in Neural Information Processing Systems*, 33:17022–17033, 2020a.

Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*, 2020b.

Junhyeok Lee and Seungu Han. Nu-wave: A diffusion probabilistic model for neural audio upsampling. *arXiv preprint arXiv:2104.02321*, 2021.

Naihan Li, Shujie Liu, Yanqing Liu, Sheng Zhao, and Ming Liu. Neural speech synthesis with transformer network. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 6706–6713, 2019.

Jae S Lim. Signal estimation from modified short-time fourier transform. *IEEE Trans., ASSP*, 32, 1984.

Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2794–2802, 2017.

Michael McAuliffe, Michaela Socolof, Sarah Mihuc, Michael Wagner, and Morgan Sonderegger. Montreal forced aligner: Trainable text-speech alignment using kaldi. In *Interspeech*, volume 2017, pages 498–502, 2017.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

Vadim Popov, Ivan Vovk, Vladimir Gogoryan, Tasnima Sadekova, and Mikhail Kudinov. Grad-tts: A diffusion probabilistic model for text-to-speech. In *International Conference on Machine Learning*, pages 8599–8608. PMLR, 2021.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al. The kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding*. IEEE Signal Processing Society, 2011.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Ajay Ramaseshan. *Application of Multiway Methods for Dimensionality Reduction to Music*. PhD thesis, Alto University, School of Science, 12 2013.

Yi Ren, Yangjun Ruan, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. Fastspeech: Fast, robust and controllable text to speech. *Advances in neural information processing systems*, 32, 2019.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):4713–4726, 2022.

Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017.

Jonathan Shen, Ruoming Pang, Ron J Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, Rj Skerrv-Ryan, et al. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4779–4783. IEEE, 2018.

Leyuan Sheng, Dong-Yan Huang, and Evgeniy N Pavlovskiy. High-quality speech synthesis using super-resolution mel-spectrogram. *arXiv preprint arXiv:1912.01167*, 2019.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Emmanuel Vincent, Rémi Gribonval, and Cédric Févotte. Performance measurement in blind audio source separation. *IEEE transactions on audio, speech, and language processing*, 14(4):1462–1469, 2006.

Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, et al. Tacotron: Towards end-to-end speech synthesis. *arXiv preprint arXiv:1703.10135*, 2017.

Robert Weide et al. The carnegie mellon pronouncing dictionary. *release 0.7, www. cs. cmu. edu*, 2015.

Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Yingxia Shao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications. *arXiv preprint arXiv:2209.00796*, 2022.