Thesis Draft

Lukas Gardberg

October 2022

1 Abstract

Contents

2 Introduction

In recent years, the amount which humans utilize software systems for communication has increased significantly. Ranging from the early beginnings of email, text messaging, and cellular calls, to modern voice and video chats through mobile applications. However, as these technologies have progressed, so has our ability to automate parts of the interaction. Be it pre-recorded messages, selecting an option on your keypad, or automatically generating responses to emails.

One such area of innovation is spoken conversation, where modern communication software strives to make the interaction personal by developing systems that are adaptable with regards to the human on the receiving end. Such a system needs to be able to understand the chosen medium, interpret what the human is communicating, and respond appropriately. This entails a level of natural language understanding, the ability to generate a natural sounding spoken response, and is an active research area.

Efforts have historically aimed to mimic the biological systems which give rise to speech in humans, either mechanically [62], or through formant synthesis and vocal chord modelling [16]. Significant progress has also been made using rule- and expert-based systems which incorporate knowledge from experts (such as linguistis) in order to achieve natural speech [34]. In addition to this, concatenative speech synthesis has also produced promising results, and is based on stitching together pre-recorded audio clips in order to obtain a target pronunciation. However, all of these methods exhibit several drawbacks, e.g. in the form of limited flexibility, heavy reliance on human knowledge, and varying naturalness of speech [58].

In order to overcome these challenges the use of statistical parametric speech synthesis (SPSS) has grown, which aims to model the varying processes in speech using parametrized models and large amounts of data. This approach is usually divided up into three steps: text analysis, parameter prediction (acoustic model), and synthesis (vocoder). The first step involves processing the text to be spoken, which includes e.g. normalization, segmentation, and phoneme conversion, which gives more detail to the pronunciation of the text. Secondly an acoustic model is tasked with generating computer processable acoustic features, which can be seen as a compact representation of the characteristics of the audio. Examples include fundamental frequencies and spectra. Lastly a vocoder model generates audio (e.g. an audio file) based on prior acoustic features [69, 58].

Figure 1: Schematic overview of common text-to-speech components. The input text which speech is to be generated for is first analyzed to produce linguistic features. These are then fed into an acoustic model outputting acoustic features, e.g. a mel spectrogram. Lastly the acoustic features are converted into audio using a vocoder model.

Many of these ideas persisted in tandem with the rise of deep learning and the utilization of increased computational power. Neural networks have thus successfully been incorporated into all stages of SPSS

[67, 5], and research has since moved towards a fuller end-to-end approach [44] which has seen great success in several other areas [37, 10, 70]. This has popularized the term *neural TTS*, where the fully end-to-end refers to the task of generating speech directly from text, using no direct intermediate representations.

This thesis focuses on the last step of the process, the *vocoder*, and the challenges of incorporating a neural network-based vocoder into a user-facing application. Special weight is put on *diffusion-based* vocoders, which is a specific type of machine learning model which has shown great promise in several other areas.

2.1 Previous work

The first vocoder based on neural networks is WaveNet which was proposed in 2016 [44]. Contrary to spectrograms it uses linguistic features directly as inputs in order to generate audio on the sample level. It is also autoregressive, meaning that each sample is conditioned on all previous ones, and that audio therefore is generated in a circular fashion by feeding in past outputs to the input. The backbone of WaveNet is the 1-D causal convolution layer, which provides learnable filters applied only on past samples relative to the current one. In addition to this, dilation is also leveraged in order to provide the filters with a wider receptive field and achieve greater efficiency. It is worth noting that one of the main drawbacks of WaveNet and similar autoregressive models is that waveform generation is not parallelizable, therefore resulting in a longer inference time via the inherent sequential nature. Several efforts have been made in order to reduce the inference, for example through pruning or caching [29, 45].

Later progress involved developing a fuller end-to-end sequence-to-sequence approach with Tacotron [63] and Tacotron 2 [53]. The proposed encoder in the first version addressed the limitations of using complex linguistic features, and instead aimed for a transformer-based model which produces mel spectrograms from character sequences. The vocoder part involved simple Griffin-Lim reconstruction, and was later improved in the second iteration. The improvement consisted of replacing the non-parametric vocoder with a WaveNet-based neural network which could be trained separately to the encoder to produce audio from mel spectrograms.

Following the advancement of using a neural network-based vocoder, several different directions have been taken. One of the main problems addressed is creating a non auto-regressive model, and speeding up inference through parallelizable architectures. One such family of models used as vocoders are *flow-based* models. These utilize a normalizing flow which iteratively transforms a simple distribution into a complex one (i.e. the distribution of speech data) using a series of invertible mappings, and can thus achieve an approximation of a target posterior distribution [50]. For example, WaveGlow [48] is a flow-based vocoder which combines progress from Glow [33] and WaveNet to non-autoregressively generate audio above 16kHz faster than real-time.

In addition to this, generative adversarial networks (GANs) have also been utilized as vocoders [20]. The principle behind a GAN is to jointly train a generator network to generate samples from noise, and a discriminator network to determine if a candidate sample is real or synthesized. Examples include WaveGAN [15] which relies on a deep convolutional neural network as the generator, and improvements

such as GAN-TTS [9], and HiFi-GAN [35] which leverage techniques such as multiple discriminators and augmented loss functions. Such models are able to generate high fidelity audio effectively using only a single forward pass, but have been shown experimentally to be difficult to train (e.g. mode collapse) and are not able to produce likelihood estimates as a consequence of the architecture. However, they are widely considered to be the state-of-the-art in terms of inference speed.

The main model considered in this thesis is DiffWave [36], which is the first diffusion-based vocoder, characterized by it's audio-specific architecture and conditioning on spectrograms generated by an acoustic model. The model is decribed in further detail in section 3.3.1.

2.2 Problem statement

The goal of this thesis is to explore the usage of denoising diffusion probablilistic models as vocoders, what aspects are favorable or limiting in real-time applications, and how they compare to other current machine learning-based models for speech generation. To achieve this, the aim is to answer three main questions:

(TODO: Refine further)

- What are the strengths and weaknesses of denoising diffusion probabilistic models in general?
- How do suggested model improvements from other domains affect performance in the vocoding task?
- Can diffusion based vocoders be used in real-time text-to-speech applications?

3 Theory

This section introduces the major theoretical building blocks needed to understand diffusion models, as well as tools and techniques related to the vocoder problem. The processing and analysis of audio signals is covered, including the three different waveform-, spectrogram-, and mel spectrogram-representations. In addition to this, a brief introduction to machine learning and learning algorithms is given in order to provide context for the modelling and optimization methods used. Lastly the diffusion framework is presented along with current limitations and challenges, as well as a discussion on suitable evaluation methods for generative models.

3.1 Sound & Signals

The natural representation of a sound, and what our human ears have evolved to perceive, is a repetitive increase and decrease in air pressure in the form of an acoustic wave. The source of a wave can for example be a vibrating membrane whose movement sets air into motion, such as our vocal chords. Each sound possesses a certain *frequency* representing how many times the acoustic wave compresses and decompresses (oscillates) per unit of time, and is measured in Hertz (Hz). Humans have evolved to perceive sounds with a frequency between 20 Hz and 20 kHz, where sounds outside of this range are inaudible to us. Sound is most commonly recorded via a microphone in which a fixture is displaced by the propagating wave. The component inside the microphone is configured to induce an electrical signal representing the movement of the air.

In order to represent such an analog signal digitally, its value (the amount of voltage or current) is measured at a certain rate, i.e. *sampled*, where the discrete measurements are represented using quantized amplitude values such as floating point units. The rate at which the signal is measured is called the *sample rate* f_s (Hz). As a consequence of the Nyquist-Shannon sampling theorem a digital signal can only represent an analog signal without distortion (aliasing) if it is sampled with a sample rate at least double the frequency of the highest frequency in the analog signal. For example, a signal sampled at 44 100 Hz can at most adequately represent an analog signal of 22 050 Hz, making it an appropriate sample rate for human hearing. (Cite?)

Mathematically it is possible to represent an audio signal in several ways. One form suitable for theoretical analysis is to define it as a function $x : \mathbb{R} \to \mathbb{R}$, where $x(t)$ represents its amplitude at a real valued time-point $t \in \mathbb{R}$. However, in practice measured signals are not defined at infinitely many points, but are rather of limited size as a consequence of the time interval over which they were sampled being finite. We define such a real-word signal as a *vector* $\boldsymbol{x} = [x[0], \ldots, x[L-1]]^{\mathrm{T}} \in \mathbb{R}^L$ which consists of L real-valued amplitude samples indexed from 0.

3.1.1 The Short-time Fourier Transform

For the task of analyzing the frequency contents of signals such as audio, the *Fourier Transform* is a common and useful tool. Given a signal $x : \mathbb{R} \to \mathbb{R}$ its transform $y : \mathbb{R} \to \mathbb{C}$ is defined as

$$y(f) = \int_{-\infty}^{\infty} x(t) e^{-i2\pi ft}\, dt, \tag{1}$$

for a specific frequency $f \in \mathbb{R}$. The resulting function y provides amplitude and phase information of each frequency contained in x in the form of the absolute value and the argument of the resulting complex number. Intuitively the Fourier Transform can be interpreted as a measure of how correlated a signal is with a sinusoid of frequency f. Once y has been obtained, it is also possible to perform an inverse transform

$$x(t) = \int_{-\infty}^{\infty} y(f) e^{i2\pi ft}\, df \tag{2}$$

in order to recover the original signal x under certain conditions.

As earlier described, signals are often represented as sampled values at discrete time points in contrast to a real valued function. The respective transform for a discrete-time signal $\boldsymbol{x} = [x[0], x[1], \ldots, x[L-1]]^{\mathrm{T}}$ is the *Discrete Fourier transform* (DFT), which is defined as

$$y[k] = \sum_{n=0}^{L-1} x[n] e^{-i\frac{2\pi}{L}kn}, \tag{3}$$

resulting in a new complex valued vector $\boldsymbol{y} = [y[0], \ldots, y[L-1]]^{\mathrm{T}}$ where $\boldsymbol{y} \in \mathbb{C}^L$. Worth noting is that both these transforms analyze the complete signal all at once which yields a static spectrum, and can thus be ill suited for signals whose frequency content change over time. An example of such a changing signal is recorded speech, where the pronunciation of different vowels and consonants in the form of sustained and plosive sounds result in a dynamic frequency distribution.

In order to account for this variability the signal can instead be analyzed at several overlapping individual segments. This is done by *windowing* the signal with a window function w which is zero outside of a set width. In practice this function is represented as a vector $\boldsymbol{w} = [w[0], \ldots, w[L-1]] \in \mathbb{R}^L$. The resulting transform of such a windowed signal gives a representation of the frequency content of that particular segment, allowing several such windowed transforms to be combined into a time-dependent frequency distribution. This process is performed using the *Short-time Fourier Transform* (STFT), which for a signal $\boldsymbol{x}$ and window function $\boldsymbol{w}$ is defined as

$$\mathrm{STFT}\left\{\boldsymbol{x}\right\}(k, m) = Y[k, m] = \sum_{n=0}^{N-1} x[n] w[n + mH] e^{-i\frac{2\pi}{N}kn}. \tag{4}$$

Here k represents the frequency bin index, and m the time frame index. The *hop length* of the window is denoted by H which determines the distance between two adjacent windows. Note also that the number of evaluation points $N \leq L$ have to be chosen, which differs from the number of samples L in $\boldsymbol{x}$. The result of the STFT is a matrix $\boldsymbol{Y} \in \mathbb{C}^{K \times M}$ of complex numbers $Y[k, m]$ for each frequency and time

index. However, this representation is impractical for humans to analyze manually through e.g. visual inspection. One approach of reducing the complexity and obtaining a visual representation is to calculate the *spectrogram* $\boldsymbol{S}$, which is defined as the element-wise squared magnitude of the STFT matrix $\boldsymbol{Y}$, i.e.

$$S[k, m] = |Y[k, m]|^2 . \tag{5}$$

Each resulting element $S[k, m]$ is thus a real value, which enables $\boldsymbol{Y}$ to be displayed as an image. One important aspect of this simplification is that when the entries of the complex matrix $\boldsymbol{Y}$ are squared, some phase information of the signal is lost (the arguments of the complex numbers), and a lossless inverse transform is no longer possible. This leaves the power spectral density, which can be seen as a representation of how the energy is divided between the different frequencies over time. An example of a speech recording and its spectrogram can be seen in figure 2.

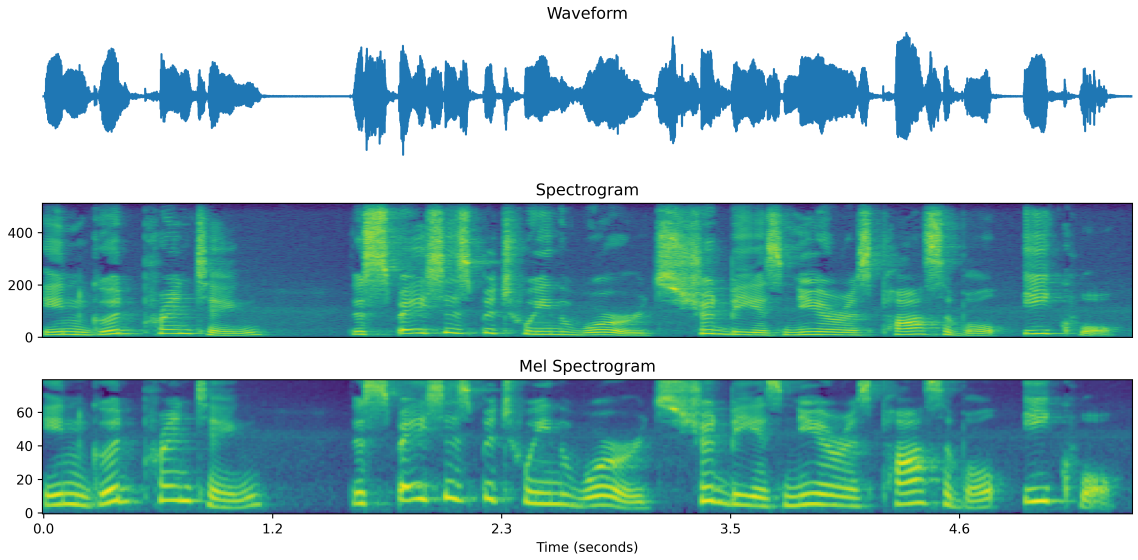

Figure 2: Waveform, Spectrogram, and Mel Spectrogram representation of the spoken sentence *"In good printing, the spaces between words should be as near as possible equal"*. The x-axis denotes the time in seconds, and the y-axis denotes the frequency- and mel-bins for the spectrogram and mel spectrogram respectively. Note that the mel spectrogram contains fewer bins than the spectrogram as a result of the choice of 80 mel filters, but that we observe a more detailed resolution for the speech components of the signal in the form of overtones. A brighter point represents a larger magnitude. Spectrograms were obtained identically as described in section x.

One notable aspect of the STFT is that the resolution of $\boldsymbol{Y}$ in both time and frequency is related to the choice of window $\boldsymbol{w}$. For a wider window a greater frequency resolution is achieved, meaning that it becomes easier to distinguish between components which are close in frequency. However, a wider window results in worse time resolution, as fewer windows are needed to cover the entire signal. Conversely a narrow window gives better time resolution as the DFT is calculated at more time steps, but a worse frequency resolution. This trade off makes the choice of $\boldsymbol{w}$ important, and has given rise to several

methods which aim to analyze the signal at multiple time- and frequency resolutions. (cite?)

TODO: Add MFCC definition

3.1.2 The Mel Spectrogram

When analyzing a signal using the STFT the resulting complex matrix $\boldsymbol{Y}$ contains information about its frequency content for each *linearly spaced* frequency k. However, it is widely understood that the human perception of the distance between two different frequencies is not linear, as experiments have shown that this distance gets larger for higher frequencies [56]. This is characterized by the *mel-scale*, which can be used to transform a linear frequency scale into one that is adjusted for human hearing. Transforming a spectrogram according to this scale results in a *mel spectrogram*, and provides a higher frequency resolution for the range of frequencies in which human speech is centred. Such a transform which maps a frequency f in Hertz into *mels* is defined as

$$f \longmapsto 2595 \log_{10} \left(1 + \frac{f}{700}\right). \tag{6}$$

Also worth noting is that the transform compresses higher frequencies and expands lower ones through it's logarithmic transform. This can also be seen as a representation which is closer to how the brain processes sound, and is thus better suited for representing human speech.

In practice a mel spectrogram is obtained using a mel-frequency filter bank. For each time step in the spectrogram, each mel coefficient is obtained by multiplying the corresponding spectrum with a band-pass filter, such as a triangle window. For e.g. 80 filters this results in 80 mel components, corresponding to the y-axis of the mel spectrogram in figure 2. The band-pass filters are chosen to be *linearly spaced in the mel-scale*, meaning that frequencies are equal distances apart in terms of how they are perceived by humans. The process of transforming a magnitude spectrogram $\boldsymbol{S}$ of size $K \times M$ into a mel spectrogram $\boldsymbol{S}_{\text{mel}}$ of size $K_{\text{mel}} \times M_{\text{mel}}$ can be defined as $\mathcal{M} : \boldsymbol{S} \mapsto \boldsymbol{S}_{\text{mel}}$. An example of a mel spectrogram can be seen in figure 2, and for the specific mel spectrogram settings used in experiments, see section x.

Overall the mel spectrogram provides a natural way of representing human speech in a format which is processable by computers. Importantly it is a more compressed representation of the audio. For example, the waveform in figure 2 contains around 120 000 samples, while the corresponding mel spectrogram shown only contains around 38 000, which is roughly three times as few. In practice the actual size in memory would depend on other aspects such as floating points precision, but it nonetheless highlights one of the main favourable qualities of the representation.

3.1.3 Phase Reconstruction Problem

Say a spectrogram representation of a signal has been obtained, but the original signal is no longer available. How would one go about recreating the original signal? This is called the *phase reconstruction problem*, which refers to the fact that if the original phase of the signal is reconstructed the signal can easily

be recovered through an inverse transform. As described in sections 3.1.1 and 3.1.2, signal information is discarded both in the complex- to real-valued spectrogram computation, as well as the mel-frequency filter bank transformation. Considering the problem of recovering a signal $\boldsymbol{x}$ from its mel spectrogram, the mel spectrogram is first obtained through an STFT, a magnitude transform, and a transform $\mathcal{M}$. The heart of the phase reconstruction problem lies in being able to invert this process using a *vocoder* in order to recover a signal $\tilde{\boldsymbol{x}} \approx \boldsymbol{x}$. This series of transformations is depicted in figure 3. Note that a mel spectrogram is obtained directly from the available ground truth signal when training the vocoder, whereas during inference for text-to-speech it would be provided by an acoustic model as in figure 1.

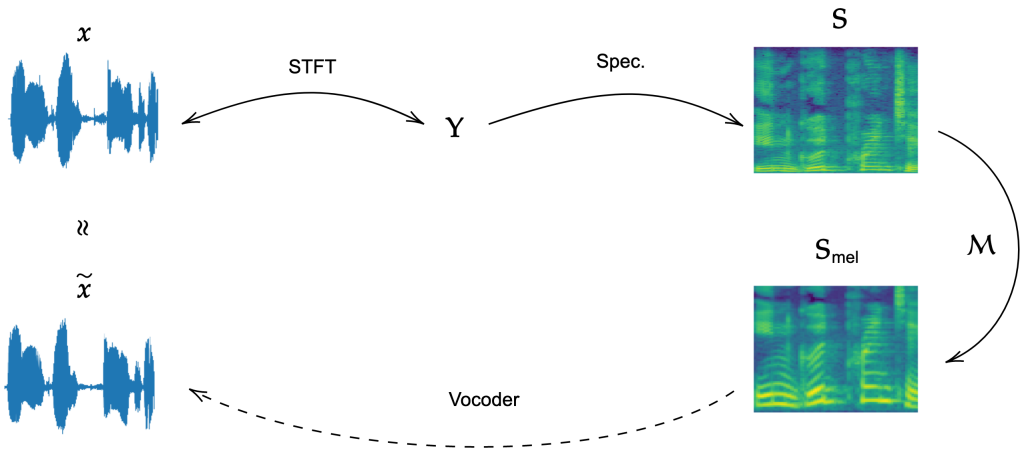

Figure 3: The process of obtaining a mel spectrogram $\boldsymbol{S}_{\text{mel}}$ from an original signal $\boldsymbol{x}$, and recreating it using a vocoder. The signal is first transformed using a Short-time Fourier Transform, which is depicted using a bidirectional arrow to emphasize its invertibility. A spectrogram $\boldsymbol{S}$ is then obtained though a magnitude transform, followed by a mel spectrogram $\boldsymbol{S}_{\text{mel}}$ via a transform $\mathcal{M}$. These transforms are shown as unidirectional since they are non-invertible. Lastly the vocoder performs the inverse mapping to obtain $\tilde{\boldsymbol{x}}$. This procedure is followed during training of the vocoder model in order to generate training data, whereas during inference the mel spectrogram is generated by an acoustic model.

At a first glance modelling such a transformation may seem impossible through a lack of information. However, previous research has shown both experimentally and theoretically that the problem is tractable by e.g. utilizing a number of underlying connections between the phase and the magnitude of the spectrogram, or learning the mapping via large amounts of data [30, 1, 22, 6]. (Lägga till något om att träna på spectrogram genererade från ground truth data vs från en acoustic model?)

Historically there have been many different approaches to solving the phase reconstruction problem. It is evidently of special interest for text-to-speech applications because of the need to transform the acoustic features generated by an acoustic model into audio. The main focus in this thesis is on learning the inverse mapping needed for reconstruction through a statistical model trained on a large amounts of mel spectrogram and audio pairs. However, popular preceding methods were often data-agnostic and instead exclusively relied on numerical optimization or instrinsic assumptions about the signal, which here can

be of great use as a baseline [60, 21].

3.1.4 Griffin-Lim reconstruction

The Griffin-Lim vocoder is a traditional approach used to recover an audio signal from the magnitude of its STFT. More specifically, it is based on reducing a simple mean squared error between the magnitude of the STFT of the estimated signal $\tilde{\boldsymbol{x}}$ and the original magnitude representation, i.e the spectrogram. The number of iterations can manually be adjusted to achieve a better reconstruction. The algorithm was originally proposed in 1984, has since then been used as a baseline in several previous vocoder studies [63, 48], and provides a simple and non-parametric way of solving the phase reconstruction problem [21, 2].

3.2 Learning Paradigms & Generative Models

Some of the first attempts to solve the phase reconstruction problem consisted of methods which mainly relied on signal assumptions and numerical optimization, and more importantly only considered a single signal at a time. The alternative emergent approach is to instead attempt to learn the mapping by using information from a large amount of samples in the hope of achieving a solution which performs better in general. The process of creating a statistical model to solve a task by observing data is one way to define *machine learning*. It has become especially prominent today thanks to more data being readily available, and an increased capability of performing large-scale computation, especially in parallel.

Historically the most common learning framework is *supervised learning*. Given a set of pairs of samples (e.g. audio signals) $\boldsymbol{x} \sim q$ drawn from a data distribution q and labels y, the goal is to learn the probability of a sample $\boldsymbol{x}$ having label y, i.e. $q(y \mid \boldsymbol{x})$. The term "supervised" is used because the ground truth label y needs to be assigned by a human, essentially *supervising* the learning. Because the data of interest often times is high dimensional and possesses a complex structure it is impossible to obtain $q(y \mid \boldsymbol{x})$ explicitly. Therefore an approximate model $q_\theta(y \mid \boldsymbol{x})$ is used to estimate $q(y \mid \boldsymbol{x})$, where θ are the parameters defining the model to be learned. A common goal of supervised learning is to be able to classify samples $\boldsymbol{x}$ into one of several categories, such as if an image represents a cat or a dog, or if an audio recording consists of a dog barking or a cat meowing [19].

Related is the task of *unsupervised learning* which aims to approximate the original underlying data distribution $q(\boldsymbol{x})$ which the observed data stems from. This is similarly done through an approximate model $q_\theta(\boldsymbol{x})$ which estimates $q(\boldsymbol{x})$. The goal of unsupervised learning is often to be able to generate new, unseen samples from $q(\boldsymbol{x})$ using $q_\theta(\boldsymbol{x})$, and is the specific sub-field especially considered in this thesis, coined *generative models*. In certain cases it may be desirable to obtain explicit probability values $q(\boldsymbol{x})$, e.g. for comparing the negative log likelihood between models (see section 3.2.2). However, in the context of generative models weight is rather put on being able to generate new "realistic" samples which follow the original probability distribution as closely as possible, allowing $q(\boldsymbol{x})$ to only be available implicitly [17]. Evaluating such a model without access to $q(\boldsymbol{x})$ is further discussed in section 3.4.

Closely tied to unsupervised learning is *self-supervised learning*. Instead of using a human-made learning signal y, a model is trained by withholding some information from the data, which then is to be predicted using the remaining available data. In this way the learning signal instead comes directly from the data set, which is why the model can be considered to supervise itself. For example, both language models and speech recognition systems can be trained through *input masking* where the model is tasked with predicting a removed part of a sequence given the preceding data, such as a sentence or a waveform. However, there is not a clear boundary between self-supervised and unsupervised learning, and the chosen vocoder model framework can therefore both be considered to be unsupervised and self-supervised [8, 14].

In the case when labels y exist, it is also possible to create a generative model which estimates $q(\boldsymbol{x} \mid y)$, i.e. conditions on the label. In the case of generating cat and dog sounds, one could imagine the model to be conditioned to either generate audio of a dog or a cat by feeding the label into the model during sample generation. The idea of learning both tasks jointly and guiding the sample generation via a label is that the model can learn patterns common to both tasks. In practice the information a generative model is conditioned on does not have to be a label or a category, and can just as well be any other type of data which provides the model with an informative signal which limits the range of data considered. In the context of vocoders and the phase reconstruction problem, the vocoder is conditioned on acoustic features such as a mel spectrogram when performing the task of generating speech, which one can imagine limits the number of plausible audio sources significantly.

3.2.1 Neural Networks

In order to obtain a good enough approximation $q_\theta(\boldsymbol{x}) \approx q(\boldsymbol{x})$ the first step is to choose a function q_θ which is able to capture the complexity of q adequately. For example, if we know that $\boldsymbol{x}$ is drawn from a normal distribution, i.e. $q(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ it is reasonable to choose a model $q_\theta(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_\theta, \boldsymbol{\sigma}_\theta^2)$. In fact in this case one can show that we can obtain an approximation which is arbitrarily close to q with enough data (cite?). However, that is not true for any type of distribution q, and it becomes significantly harder to come up with or even reason about what type of distribution a sample stems from when the data is more complex. Loosely speaking, we can say that the *capacity* of the chosen model q_θ is limited since it is restricted to modelling normally distributed data. Conversely we would say that a model which can capture more complex data distributions, i.e. approximate more complex functions, has a higher capacity. One model with such capability is the *neural network*.

A neural network consists of a combined number of functions which are chained together to produce an output. The main idea is to obtain a function with higher capacity by combining several non-linear functions with lower capacity. One such basic function is the perceptron. Given an input vector $\boldsymbol{x} = [x_0, \ldots, x_{L-1}]^{\mathrm{T}}$, which e.g. can represent an audio signal where each element $x_k = x[k]$ (as defined in section 3.1), a perceptron takes one or more inputs x_k, performs an affine transformation, and then passes the combined value through an element-wise non linear *activation function* φ. A perceptron k is parametrized by its multiplicative weights $\boldsymbol{w}_k$ and additive bias b_k. The output h_k can be written as

$$h_k = \varphi\left(\boldsymbol{w}_k^T \boldsymbol{x} + b_k\right), \quad \theta = \{\boldsymbol{w}_k, b_k\}. \tag{7}$$

Several of these perceptrons can be stacked and their output combined in order to form a *layer*, where the number of perceptrons in the layer is called the *width*. The layer will then have an output for each perceptron, which can be written in vector form as

$$\boldsymbol{h} = \varphi\left(\boldsymbol{W}^T\boldsymbol{x} + \boldsymbol{b}\right), \quad \theta = \{\boldsymbol{W}, \boldsymbol{b}\}, \tag{8}$$

where each perceptron in the layer takes in all inputs $\boldsymbol{x}$ from the previous layer, making it fully connected. Here $\boldsymbol{W}$ is an $L \times P$-matrix, where P is the number of outputs h_k, where each row k represents the multiplicative weights of perceptron k. The output $\boldsymbol{h}$ and bias $\boldsymbol{b}$ are both column vectors of length P. For example, the outputs of the "hidden layer" $\boldsymbol{h}$ of the network shown in figure 4 is calculated as

$$\underbrace{\begin{bmatrix} h_0 \\ h_1 \\ h_2 \end{bmatrix}}_{\boldsymbol{h}} = \varphi\left(\underbrace{\begin{bmatrix} w_{0,0} & w_{0,1} \\ w_{1,0} & w_{1,1} \\ w_{2,0} & w_{2,1} \end{bmatrix}}_{\boldsymbol{W}^T}\underbrace{\begin{bmatrix} x_0 \\ x_1 \end{bmatrix}}_{\boldsymbol{x}} + \underbrace{\begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix}}_{\boldsymbol{b}}\right), \tag{9}$$

where $w_{k,j}$ corresponds to the weight of perceptron k on input component j. Consequently it is possible to both extend the neural network by increasing the width of the hidden layer, or adding more hidden layers (i.e. increasing the depth). Both the depth and width extension provide the network with more learnable parameters, and thus increase its capacity. Creating such a network with several layers is often called a *multilayer perceptron* (MLP) and can be considered the cornerstone of the field *deep learning*, which deals with MLPs of considerable depth.

When training such a network the dimensionality of the input is often decreased with the depth as a consequence of the architecture. This results in the network being forced to achieve more effective representations of the input data if it is to succeed at the task its being train to perform (commonly identified as *representation learning*). These intermediate lower-dimensional representations are often called *embeddings*, and can in fact be used to perform other tasks as well. For example, a language model might have been trained in a self-supervised fashion to generate text, and through this learned to generate effective text embeddings. These can then be utilized to e.g. classify text by using them as inputs to another model.

The neural network or MLP architecture provides an effective way to adjust the capacity of our approximating function. In fact it has been shown that neural networks are universal approximators, meaning that they can approximate (almost) any function or probability distribution arbitrarily well provided either unlimited depth or width [25, 41]. This flexibility is one of the main reasons for their widespread usage. However, the process of actually obtaining such an approximation is an entire challenge of its own, and has arguably been one the main areas of research focus within the field in recent years.

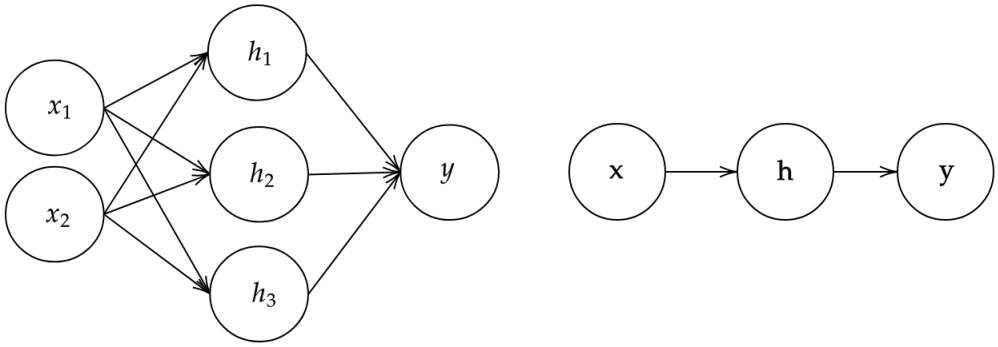

Figure 4: A depiction of a simple neural network with two inputs, a single hidden layer with a width of three, and a single output. Each hidden value h_k is calculated via equation (7). The right side of the figure shows the network written in vector form.

Even though the "vanilla" neural network architecture can in theory approximate almost any function, doing so effectively using a limited number of weights is hard, as well as obtaining an approximation which generalizes to unseen data. One approach of easing the learning process is to adapt its architecture to the modality of the data. This has for example been done for images via the *convolutional neural network* (CNN) [39]. Its structure is based on the assumption that the relationship between samples (e.g. pixels) is highly local, and thus only considers a subset of the input at a time compared to regular fully connected networks. This subset is often times in the form of a filter which is moved across the signal and used to calculate an element-wise product at each position. In the image case the filter can be of shape $D \times H \times W$, where D, H and W represent the size of the depth, height, and width of the filter. For a one dimensional signal it usually of the shape $1 \times F$, where F represents the length of the filter. Its strength lies in the fact that the filters consist of learnable weights, which allows the network to learn what types of features should be recognized, compared to a fixed, hand-made convolutional filter. It is also especially powerful since chaining together several convolutional layers allows the network to learn a hierarchy of features which can aid the learning process additionally [19].

Because the same filter is used at several positions in the input signal, the network achieves a higher parameter efficiency. However, it can still be costly to increase the size of the filter in order to be able to capture "larger" features. One way of mitigating this is through *dilated* convolutional filters, which do not consider a dense subset of samples, but instead consist of repeated skips. This results in a less fine grained filter with "holes" in it, but allows the network to achieve a wider receptive field using fewer parameters [66, 44]. In addition to this there are many improvements and tricks which have emerged to effectively train neural networks, including residual connections, different activation functions, layers, and architectures. For more specific information regarding the techniques used in the mentioned models we refer the reader to the respective papers [36, 40, 44, 58].

3.2.2 Learning Algorithm

In the context of unsupervised learning, once a model q_θ has been chosen we want to be able to adjust its parameters so that it better approximates our target distribution q_{data}, as well as be able to evaluate the quality of this approximation. Assume we have a *training set* of observed samples $\mathbb{X} = \{\boldsymbol{x}^{(i)}\}_{i=1}^{N} \subset \mathbb{N} \times \mathbb{R}^L$ where each data point stems from our empirical data distribution, i.e. $\boldsymbol{x}^{(i)} \sim q_{\text{data}}$. One way to define a model with optimal parameters θ^* is through the maximum likelihood estimation of θ, which is defined as

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{N} q_\theta\left(\boldsymbol{x}^{(i)}\right) = \underset{\theta}{\operatorname{argmin}} \left[-\mathbb{E}_{\boldsymbol{x}\sim q_{\text{data}}} \log q_\theta(\boldsymbol{x})\right]. \tag{10}$$

This can be interpreted as minimizing the dissimilarity between our chosen model distribution q_θ and the observed data distribution q_{data}, or equivalently the Kullback-Leibler (KL) divergence.

The *Kullback-Leibler divergence* provides a way to measure how different two distributions P and Q over the same variable $\boldsymbol{x}$ are, and is defined as

$$D_{\text{KL}}(P \mid Q) = \mathbb{E}_{\boldsymbol{x}\sim P}\left[\log \frac{P(\boldsymbol{x})}{Q(\boldsymbol{x})}\right] = \mathbb{E}_{\boldsymbol{x}\sim P}\left[\log P(\boldsymbol{x}) - \log Q(\boldsymbol{x})\right]. \tag{11}$$

The KL divergence is often thought of as a distance measure between distributions. However, it is not symmetric, and therefore cannot be considered one (hence the name *divergence*) [19].

From our maximum likelihood estimation we can define the *loss function*

$$L(\boldsymbol{x}, \theta) = -\log q_\theta(\boldsymbol{x}), \tag{12}$$

which measures the cross entropy between our model probability distribution and the empirical probability distribution defined by $\mathbb{X}$. This means that changing θ to reduce L will result in a model which is closer to our target distribution. Another term for (12) is the *negative log likelihood* (NLL). The average loss $\mathcal{L}(\theta)$ for the entire training set $\mathbb{X}$ and a specific set of weights θ can be calculated as

$$\mathcal{L}(\theta) = \mathbb{E}_{\boldsymbol{x}\sim q_{\text{data}}} L(\boldsymbol{x}, \theta) = \frac{1}{N}\sum_{i=1}^{N} L(\boldsymbol{x}^{(i)}, \theta). \tag{13}$$

The model is trained through gradient descent which updates the parameters of the model by taking a step in the negative direction of the gradient of $\mathcal{L}$. However, for a large data set it can be costly to compute the gradient w.r.t. each data point in the training set. One solution is to instead use a random subset of the training set in each calculation, essentially estimating the expectation in equation (13). This method is called *stochastic* gradient descent, whose inherent randomness has actually shown to be favourable in several ways to the learning. Forming such a batch subset $\mathbb{B} = \{\boldsymbol{x}^{(i)}\}_{i=1}^{B} \subset \mathbb{X}$ by randomly choosing $B << N$ samples the approximate training set loss $\mathcal{L}_b \approx \mathcal{L}$ can be calculated as

$$\mathcal{L}_b(\theta) = \frac{1}{B}\sum_{i=1}^{B} L(\boldsymbol{x}^{(i)}, \theta), \tag{14}$$

which can be used to form the approximate gradient

$$\nabla_\theta \mathcal{L}_b(\theta) = \frac{1}{B} \nabla_\theta \sum_{i=1}^{B} L(\boldsymbol{x}^{(i)}, \theta). \tag{15}$$

The gradient descent algorithm then updates the weights of the model through

$$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_b(\theta) \tag{16}$$

where the *learning rate* η defines the size of each step taken. In practice more sophosticated optimization algorithms such as Adam are often used, however these are intentionally left out due to them being out of scope for this thesis. For more detailed information we refer to [32, 52].

In (15) we defined the gradient of the loss for a batch of data points. In practice performing this gradient calculation is a complicated process and done using an algorithm called *backpropagation*. It calculates and stores gradients of the loss function with respect to each parameter in the network in the forward pass, which then can be used by the optimizer for parameter updates. Because the output of the network consists of several operations composed together, extracting the gradient of a specific parameter becomes a problem of calculating several partial derivatives combined through the chain rule. For example, for the loss of a single example $\boldsymbol{x}^{(i)}$ the gradient of a weight $w_{1,1}$ in the first layer of the network shown in figure 4 is calculated as

$$\nabla_{w_{1,1}} L(\boldsymbol{x}^{(i)}, \theta) = \frac{\partial L}{\partial y} \frac{\partial y}{\partial h_1} \frac{\partial h_1}{\partial w_{1,1}}. \tag{17}$$

This expression of course grows in complexity as the size and architecture of the network changes. In practice gradient tracking and calculation is done automatically using a machine learning programming framework such as PyTorch [46] which implements the backpropagation algorithm, as well as is able to utilize parallelization in order to speed up calculations. This is the method which the majority of modern neural network-based models are trained today, and is also what will be used in this thesis.

Having now established the theoretical basis for unsupervised models and the tools used to optimize them, the coming sections will focus on one family of unsupervised models called *denoising diffusion probabilistic models* (or for short: *diffusion models*). This model framework provides an effective way of generating novel samples from a target data distribution q, but as they are a relatively new approach and still an active research area they still exhibit a number of problems which limit their performance. The characteristics of these limitations will be covered, especially in the context of the vocoding task, as well as a range of prior work which aim to address them.

3.3 Diffusion models

Denoising diffusion probabilistic models were originally proposed in [54], and are a class of unsupervised generative models which aim to enable sampling from a data distribution q_{data}. The main idea behind diffusion is to model the transition between such a complex real-world distribution q_{data} and a simple latent prior p_{latent} by gradually diffusing from the first to the second, which is done by sequentally adding

noise to data from the target distribution. This problem can be framed from several perspectives, and is connected to topics such as non-equilibrium thermodynamics [54], simulated annealing [42], and score matching [55].

The base of the process consists of a Markov chain which gradually adds Gaussian noise to the original data until it is transformed into a Gaussian prior. The Markov chain is defined to have $T+1$ states, where a sample $\boldsymbol{x}$ in state $t \in [0, T]$ is denoted as $\boldsymbol{x}_t$. Given an original data distribution q_{data} and samples $\boldsymbol{x}_0 \sim q_{\text{data}}(\boldsymbol{x}_0)$ the *forward diffusion process* can be defined as

$$q(\boldsymbol{x}_{1:T} \mid \boldsymbol{x}_0) = \prod_{t=1}^{T} q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}), \tag{18}$$

$$q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t \mathbf{I}) \tag{19}$$

where for a number of chosen diffusion steps T, $\boldsymbol{x}_{1:T} = \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$ are the sequential latent samples obtained by gradually adding noise to $\boldsymbol{x}_0$, and $\{\beta_t\}_{t=1}^{T}$ a chosen *variance schedule*, which determines the variance of the noise added in each step. For the vocoding task the data $\boldsymbol{x}_0 \in \mathbb{R}^L$ is a real-valued audio signal of length L.

We then define a *reverse distribution* $p_\theta(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)$ which determines the transition probabilities of the Markov chain in reverse, i.e. going from p_{latent} to q_{data}. Starting with $p_{\text{latent}}(\boldsymbol{x}_T) = \mathcal{N}(\boldsymbol{x}_T; \mathbf{0}, \boldsymbol{I})$, the reverse process is defined as

$$p_\theta(\boldsymbol{x}_{0:T}) = p_{\text{latent}}(\boldsymbol{x}_T) \prod_{t=1}^{T} p_\theta(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t), \tag{20}$$

$$p_\theta(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t), \boldsymbol{\Sigma}_\theta(\boldsymbol{x}_t, t)). \tag{21}$$

Here $\boldsymbol{\mu}_\theta$ and $\boldsymbol{\Sigma}_\theta$ denote learned means and variances which determine the characteristics of the denoising done in each step. Considering this framework on a high level, the aim is to obtain a model p_θ which can generate novel samples by starting with latent noise $\boldsymbol{x}_T$ and sequentially performing de-noising steps for each t by drawing from $p_\theta(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)$ until $\boldsymbol{x}_0$ is obtained (see Figure 5).

It is desirable to be able to draw a sample directly from any of the intermediate distributions in the forward process for a specific step t, compared to having to sequentially add noise to the initial sample $\boldsymbol{x}_0$. As shown by [24] this can be done by first defining $\alpha_t = 1 - \beta_t$, and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$. Then, thanks to the nice properties of the Gaussian, the distribution of a sample $\boldsymbol{x}_t$ in the forward process can be written as

$$q(\boldsymbol{x}_t \mid \boldsymbol{x}_0) = \mathcal{N}\left(\boldsymbol{x}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)\boldsymbol{I}\right), \tag{22}$$

$$\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}), \tag{23}$$

enabling a way to directly obtain $\boldsymbol{x}_t$ via (23). In this context we set

$$\boldsymbol{\Sigma}_\theta(\boldsymbol{x}_t, t) = \sigma_t^2 \boldsymbol{I}, \quad \sigma_t^2 = \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t}\beta_t \tag{24}$$

as originally proposed. This discards $\boldsymbol{\Sigma}_\theta$ as a learnable parameter and instead lets it be determined by our choice of β_t.

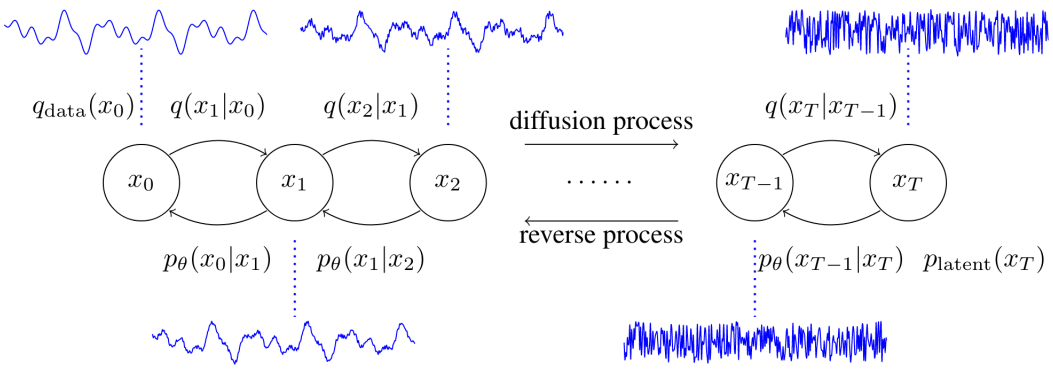

Figure 5: The reverse and forward diffusion process for a given audio signal. Through T diffusion steps a sample from a prior q_{data} is transformed into a sample from a distribution p_{latent}. This process is then to be reversed through a series of reverse transitional probabilites $p_\theta(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)$. Image adopted from [36].

For our choice of model p_θ a learning objective, or loss function, is needed in order to approximate the backwards process. Similar to an autoencoder [19], our loss function can be chosen as the variational upper bound of the negative log likelihood since optimizing the NLL directly is intractible, i.e.

$$\mathbb{E}[-\log(p_\theta(\boldsymbol{x}_0))] \leq \mathbb{E}_q \left[-\log p_\theta(\boldsymbol{x}_t) - \sum_{t \geq 1} \frac{p_\theta(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)}{q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1})} \right] =: -L_{\text{ELBO}}. \tag{25}$$

(Longer derivation of the bound needed? Talk about why we keep the variance constant?)

Following [24], they found that re-parametrizing and learning to approximate the noise $\boldsymbol{\varepsilon}$ by $\boldsymbol{\varepsilon}_\theta$ instead of the mean $\boldsymbol{\mu}_\theta$ led to better performance (expand derivation?). Thus, the training objective can be simplified to

$$L_{\text{simple}} = \mathbb{E}_{t, \boldsymbol{x}_0, \boldsymbol{\varepsilon}} \|\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_\theta(\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\varepsilon}, t)\|_2^2, \quad \boldsymbol{\varepsilon}_\theta : \mathbb{R}^L \times \mathbb{N} \to \mathbb{R}^L. \tag{26}$$

During training the noise level of each sample is determined by $t \sim p_t$, which e.g. can be set to $p_t = \mathcal{U}(0, T)$. Furthermore, each sample is drawn as $\boldsymbol{x}_0 \sim q_{\text{data}}(\boldsymbol{x}_0)$, and the additive noise according to $\boldsymbol{\varepsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. This means that for a random time step t, the model is to predict the noise $\boldsymbol{\varepsilon}$ which was added to a sample $\boldsymbol{x}_{t-1}$ to obtain $\boldsymbol{x}_t$, using only $\boldsymbol{x}_t$. In other words, the model's objective is to be able to take a reverse step in the forward diffusion process.

The training and sampling algorithms can be defined as follows:

Algorithm 1 Training algorithm

repeat
  $\boldsymbol{x}_0 \sim q_{\text{data}}(\boldsymbol{x}_0)$
  $t \sim p_t$
  $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
  $\mathcal{L} \leftarrow \|\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_\theta(\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\varepsilon}, t)\|^2$
  Update parameters θ with $\nabla_\theta \mathcal{L}$
until converged

Algorithm 2 Sampling algorithm

$\boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
for $t = T, ..., 1$ **do**
  $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ if $t > 0$, else $\boldsymbol{z} = \mathbf{0}$
  $\boldsymbol{x}_{t-1} \leftarrow \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\varepsilon}_\theta(\boldsymbol{x}_t, t)\right) + \sigma_t \boldsymbol{z}$
end for
return $\boldsymbol{x}_0$

3.3.1 Diffusion-based vocoders

As the training and sampling algorithm for diffusion models are necessarily different, the way such a model performs training and sampling on the phase reconstruction problem also differs. As defined in Algorithm 1, the model is trained by randomly picking a step in the denoising process, and taking a gradient step based on the model's prediction of the noise added to the sample in the previous step, given the mel spectrogram. During sampling, T sequential denoising steps are taken by the model in order to obtain a target audio signal, given the starting noise and mel spectrogram. This means that the sampling algorithm performs the mapping

$$\textbf{Algorithm 2}: (\boldsymbol{x}_T, \boldsymbol{S}_{\text{mel}}) \longmapsto \boldsymbol{x}_0, \tag{27}$$

where $\boldsymbol{x}_0$ should approximately follow the target data distribution q_{data}. The sampling algorithm thus aims to solve the originally formulated vocoding problem (3.1.3), with the addition of a starting noise $\boldsymbol{x}_T$.

One of the first diffusion-based vocoders is DiffWave [36], which builds on the originally improved diffusion model for image generation [24]. The model is trained to denoise a waveform given the step $t \sim \mathcal{U}(0, T)$ in the diffusion process, effectively predicting the noise $\boldsymbol{\varepsilon}$ added to a waveform $\boldsymbol{x}_{t-1}$ using a neural network $\boldsymbol{\varepsilon}_\theta$. The loss optimized is a variant of the evidence lower bound presented in (26), and is defined as

$$\mathcal{L}(\theta) = \mathbb{E}_{t,\boldsymbol{x}_0,\boldsymbol{\varepsilon}}\|\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_\theta(\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\varepsilon}, t, \boldsymbol{S}_{\text{mel}})\|_2^2, \qquad \boldsymbol{\varepsilon}_\theta : \mathbb{R}^L \times \mathbb{N} \times \mathbb{R}^{m \times n} \to \mathbb{R}^L, \tag{28}$$

and trained through stochastic gradient descent similar to the originally proposed diffusion model framework. Note that $\boldsymbol{\varepsilon}_\theta$ now additionally conditions on $\boldsymbol{S}_{\text{mel}} \in \mathbb{R}^{m \times n}$.

The neural network $\boldsymbol{\varepsilon}_\theta$ is non-autoregressive, which means that generating an audio sample of length L only requires $T << L$ forward passes. The architecture stems from previous models applied to audio source separation [49], which process the input and output audio with 1-dimensional convolutions. In addition to this it utilizes a set of residual layers involving bidirectional dilated convolutions, which results in a wide receptive field and allows the model to use information both before and after the considered sample. In addition to this, a diffusion step embedding is also utilized in order to provide the network

with information of which diffusion step to consider, inspired by the transformer [61]. Furthermore, the conditioning mel-spectogram is upsampled using transposed convolutions and combined with the output of the bidirectional dilated convolution. Lastly the outputs of each residual layer is combined and used to predict the output with size L.

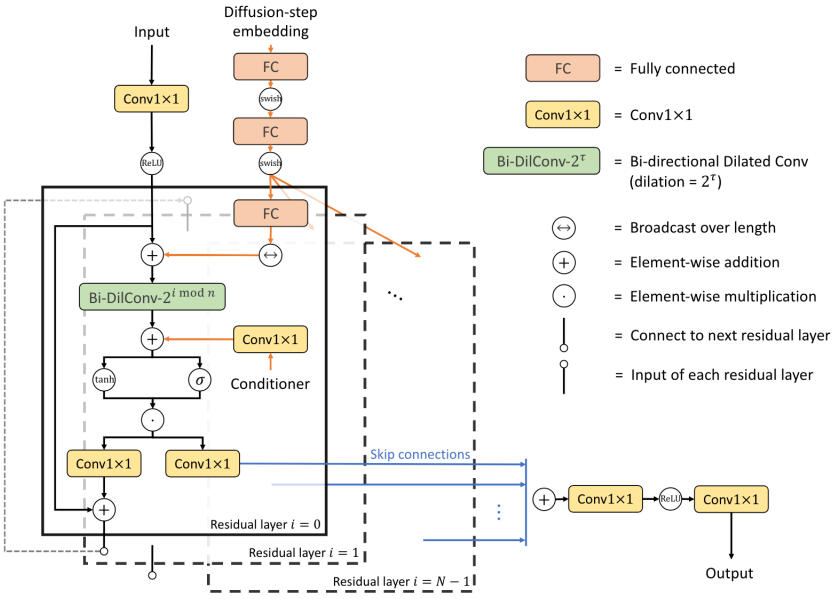

Figure 6: Neural network architecture of DiffWave [36] which models $\boldsymbol{\varepsilon}_\theta : \mathbb{R}^L \times \mathbb{N} \times \mathbb{R}^{m \times n} \to \mathbb{R}^L$. The conditioner used is in the form of a mel spectrogram S_{mel}.

The authors note that DiffWave outperforms WaveNet-based vocoders, but is still slower than most flow- and GAN-based models. They especially emphasize the need to reduce the inference time in order to be able to apply it to practical applications, among other things. The following section concerns several such limitations of diffusion models from the perspective of the vocoder. The main focus is on improvements of the diffusion model framework, as opposed to the chosen neural network architecture.

3.3.2 Limitations

It has been experimentally shown that with enough training iterations diffusion-based vocoders are able to generate high quality audio. However, their sampling process requires several forward passes which results in considerably longer inference times compared to other competitive models. Thus one of the main concerns is to be able to reduce the number of denoising steps while maintaining audio quality. Because of this, there are several aspects which affect both the quality and speed of the generation process which need to be considered [2].

Variance schedule

A central aspect to the sampling algorithm is the variance schedule β_t which determines the weighting

of the ground truth audio $\boldsymbol{x}_0$ and noise $\boldsymbol{\varepsilon}$ during training, as well as the weight of the subtractive noise $\boldsymbol{\varepsilon}_\theta$ during sampling (Algorithm 1 & 2). For image generation [24] chose the schedule to be $T = 1000$ linearly spaced steps $\beta_i \in [10^{-4}, 0.02]$, as shown in figure x. In addition to this several other studies have adopted a linear schedule [36, 40]. However, [43] noted that such a schedule does not provide the model with meaningful examples for all t, as it tends to result in Gaussian noise too early in the process (figure of the same phenomenon for audio?). They instead proposed to use a quadratic cosine schedule in terms of the noise level $\bar{\alpha}_t$ which adds noise more slowly, according to

$$\bar{\alpha}_t = \frac{f(t)}{f(0)}, \quad f(t) = \cos\left(\frac{t/T + s}{1+s} \cdot \frac{\pi}{2}\right)^2, \quad \beta_t = 1 - \frac{\bar{\alpha}_t}{\bar{\alpha}_{t-1}}, \tag{29}$$

where $s = 0.008$. They showed that using such a schedule achieves better results on the image generation task. Worth noting is that the number of sampling steps used for image generation $(1000 - 5000)$ is much higher than the number used for vocoders $(6 - 100)$, which they hypothesize is because of a much stronger conditioning signal provided in the vocoding case in the form of a mel spectrogram. Furthermore, several other methods have been proposed to achieve an effective schedule. These include framing it as a learning problem [64, 38], using numerical solvers [28], manual design [11], and hyperparameter tuning [12].

Several prior works emphasize the importance of a well designed schedule for obtaining high quality samples, especially when using an alternative shorter schedule for inference. Furthermore, it has also been noted that the optimal schedule varies with the specific data set and model, and therefore might require problem specific tuning [38, 11]. There is currently no one-fits-all method to obtain such an optimal schedule. However, several heuristics have been proposed.

TODO: Lägg till förslag på egen schedule

In addition to choosing an effective schedule, choosing an effective *prior* has also been considered. In the original framework a unit Gaussian is adopted as the prior, representing the distribution of the final noise $\mathbf{x}_T$ obtained at the end of the forward diffusion process. This also represents the starting point of the reverse process. The authors of PriorGrad [40] investigate if it is possible to choose a better prior which is closer to the target data distribution, and in turn improve the efficiency of the process. They propose to use information from the conditioning mel spectrogram in order to derive a non-standard Gaussian prior $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.

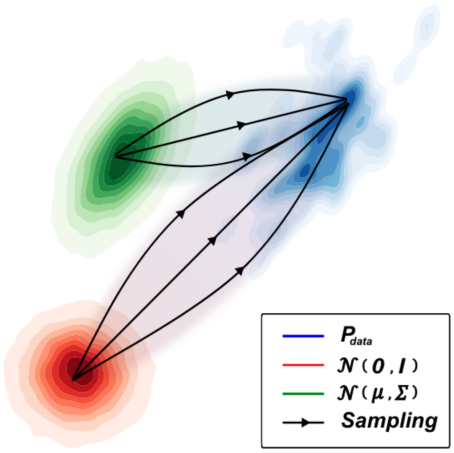

Figure 7: Intuitive depiction of starting the sampling or reverse diffusion process from a more informed prior $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ compared to the originally proposed $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ in order to obtain a sample which is closer to the real data distribution p_{data} faster [40].

In the context of a waveform-generating vocoder, $\boldsymbol{\mu}$ is chosen to be $\mathbf{0}$ as the waveform distribution is assumed to be zero mean. The authors alter the original DiffWave model to instead predict a noise $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ in each step, essentially aiming to shorten the forward and backward diffusion process.

Changing the prior results in a modified loss function which the authors derive to be

$$\mathcal{L}_{PG}(\theta) = \mathbb{E}_{x_0, \boldsymbol{\varepsilon}, t} \| \boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_\theta(\sqrt{\bar{\alpha}_t}(\mathbf{x}_0 - \mu) + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\varepsilon}, t) \|^2_{\boldsymbol{\Sigma}^{-1}}, \tag{30}$$

where $\|\boldsymbol{x}\|^2_{\boldsymbol{\Sigma}^{-1}} = \boldsymbol{x}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{x}$. The variance $\boldsymbol{\Sigma}$ is here calculated from the frame-level energy of the mel spectrogram.

In a similar fashion ResGrad [13] proposes to use a non-diffusion-based model to generate a first estimate, which is then iteratively refined by a diffusion model. This can be seen as taking a large first step closer to the target data distribution, and then taking smaller, more precise steps in order to refine the generated sample. The authors report that this method overall achieves a faster sampling speed while maintaining audio quality.

Time-step

Another aspect which affects the training and sampling algorithms is the noise step t, which was originally proposed to be sampled uniformly over $[0, T]$ [24]. However, several works have noted that this results in a noisy gradient because of the loss not being uniform over t. In addition to this, it has also been highlighted that the most effective de-noising steps occur close to $t = 0$, i.e. when the audio is close to the original sample [36, 43]. This phenomenon is shown in figure x. As a measure to counteract this, [43] propose to instead utilize importance sampling of t over $[0, T]$ weighted by the respective loss in each

time step, i.e.

$$t \sim p_t, \quad p_t \propto \sqrt{\mathbb{E}(L_t^2)}, \quad \sum_t p_t = 1. \tag{31}$$

This implies however that the loss for each time step L_t needs to be kept track of, which can for example be done through a running mean. The authors argue that such a sampling strategy can lead to more stable training and faster convergence, as well as that several such improvements cause less performance degradation when using fewer sampling steps, allowing for faster inference times. However, as the gradient is less noisy they hypothesize that less regularization is provided during training, which makes the model more likely to overfit.

The time step t is also used to condition $\boldsymbol{\varepsilon}_\theta$ to provide information regarding which step in the denoising process to perform. However, WaveGrad [11] replaced t with $\sqrt{\bar{\alpha}_t}$ and showed that similar results could be achieved. In addition to this the authors argue that conditioning on a continuous noise level makes the model more flexible to different schedules in training and inference.

(sampling + conditioning?)

Loss function

Even though the loss converges quickly after a few epochs, the authors of Grad-TTS [47] note it to be essential to continue training despite minimal loss improvements in order to achieve high quality speech. They observe that two different models with almost equal loss can produce very different samples, and hypothesize that this stems from the model's need to learn to denoise well for all $t \sim \mathcal{U}(0, T)$. (They generated spectrograms).

TODO: Expandera

(Easier to use fewer diffusion steps when sampling using e.g a conditioned spectrogram?)

3.4 Evaluation

TODO: Utöka / Skriv om?

Compared to more classic statistical problems such as classification and regression, generative models exhibit a particular challenge in how they are to be evaluated. More specifically, the performance of a generative model is generally based on the subjective human evaluation of the samples it can generate, and relied on in an ad-hoc manner to steer development. In other words, because the task we are aiming to mimic is in itself a human one, the main way to determine if it is performed well is through human judgement. This has led previous work to use a Mean Opinion Score (MOS) to evaluate the naturalness of a set of generated samples, and is currently the most widely adopted metric [44, 36, 63]. It relies on a subjective score, typically ranging from 1 to 5, which is given to a number of chosen samples by a set of human evaluators based on their perceived "naturalness". Despite being the most widely adopted metric it still has several drawbacks in terms of cost, sample size, variability, statistical significance, and comparability, to name a few [57]. Furthermore, the need for human feedback makes it significantly

harder to effectively iterate and develop models compared to quantitative metrics, especially considering the need of avoiding bias from developers.

In the context of audio, the problem of assessing quality objectively and quantitatively is a long standing one. Less recent measurement standards such as Perceptual Speech Quality Measure (PSQM), Perceptual Evaluation of Audio Quality (PEAQ), and Perceptual Evaluation of Speech Quality (PESQ) have been widely used in telephone systems, which ultimately aim to correlate with MOS [51]. Furthermore, the idea of objectifying MOS through an approximate statistical model has since then been extended. For example, [7] trained a neural network to approximate MOS of speech signals, showing higher correlation and lower MSE of opinion scores, exceeding previous established methods. As highlighted in [59] however, a subjective MOS is still the most reliable measure for generative models, and care has to be taken when choosing and comparing quantitative measures in the context of a specific application. Below four widely adopted metrics are described which have been chosen to be used for evaluating the quality of generated speech, whereas a Mean Opinion Score is not included due to a lack of resources. To compensate for this a model-based approximation of MOS is included (Fréchet Audio Distance), as well as three quantitative frequency-domain metrics.

3.4.1 Log-mel Spectrogram Mean Absolute Error (LS-MAE)

For a synthesized waveform and a ground truth signal pair, the absolute error between their mel spectrograms $\tilde{\boldsymbol{S}}_{\text{mel}}$ and $\boldsymbol{S}_{\text{mel}}$ is calculated as

$$L_{\text{LS-AE}} = \|\tilde{\boldsymbol{S}}_{\text{mel}} - \boldsymbol{S}_{\text{mel}}\|_1, \tag{32}$$

where $\|\cdot\|_1$ is the L_1 metric for matrices, and the mel spectrograms are calculated as in **??**. The LS-MAE is then obtained as the mean absolute error over a test set containing pairs of mel spectrograms of ground truth and predicted signals. This metric was previously used by [35] and [40], and aims to provide an estimate of how well the model has converged in the mel spectrogram domain.

3.4.2 Fréchet Audio Distance

Fréchet Audio Distance (FAD) [31] is a quantitative metric which aims to evaluate the quality of generated audio. It is based on the metric Fréchet Inception Distance (FID) which has been commonly used to evaluate generative models for images. Both metrics strive to correlate with the quality score a human would give to a set of samples, for example in the form of a Mean Opinion Score. FAD does not evaluate individual audio pairs, but instead compares embeddings generated from two sets of audio samples, such as as a clean and a generated one. Embeddings are obtained by extracting the penultimate hidden activations from a large audio classification model. The idea behind the metric is that if the classifier has learned to produce effective low dimensional representations (embeddings) of audio, two similar audio clips will be located closely in the embedding space [23].

For a set of clean audio samples $\{\boldsymbol{x}^{(i)}\}$ and generated audio samples $\{\tilde{\boldsymbol{x}}^{(i)}\}$ the FAD between the two

is calculated by passing the two sets into the classifier and extracting the respective sets of embeddings $\{\boldsymbol{z}^{(i)}\}$ and $\{\tilde{\boldsymbol{z}}^{(i)}\}$. Following this two multivariate Gaussians $\mathcal{N}_c(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$ and $\mathcal{N}_g(\boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g)$ are fit to the embeddings. Lastly the Fréchet distance between the two are calculated as

$$\mathbf{F}(\mathcal{N}_c, \mathcal{N}_g) = \|\boldsymbol{\mu}_c - \boldsymbol{\mu}_g\|^2 + tr\left(\boldsymbol{\Sigma}_c + \boldsymbol{\Sigma}_g - 2\sqrt{\boldsymbol{\Sigma}_c \boldsymbol{\Sigma}_g}\right). \tag{33}$$

Here tr refers to the trace of the matrix, and $\|\cdot\|$ the Euclidian distance.[1]

3.4.3 Peak Signal-to-Noise Ratio (PSNR)

The Peak Signal-to-Noise Ratio measures the ratio between the maximum power of an original signal and its corrupting noise. This is calculated as

$$L_{\text{PSNR}} = 10 \log_{10}\left(\frac{1}{\text{MSE}}\right), \tag{34}$$

measured in dB, where a peak value of 1 is assumed. The mean squared error (MSE) is calculated in the frequency domain between the mel spectrograms $\boldsymbol{S}_{\text{mel}}, \tilde{\boldsymbol{S}}_{\text{mel}}$ of the ground truth and generated audio as

$$\text{MSE} = \frac{1}{K_{\text{mel}} M_{\text{mel}}} \sum_{i=1}^{K_{\text{mel}}} \sum_{j=1}^{M_{\text{mel}}} \left(S_{\text{mel}}[i,j] - \tilde{S}_{\text{mel}}[i,j]\right)^2 \tag{35}$$

where $S_{\text{mel}}[i,j]$ and $\tilde{S}_{\text{mel}}[i,j]$ are the individual elements of the mel spectrogram matrices, and $K_{\text{mel}} \times M_{\text{mel}}$ is the size of the mel spectrogram.

3.4.4 Multi-resolution STFT Error (MRSE)

As highlighted in section 3.1.1, there will always exist a trade-off when transforming a signal using the STFT for a chosen window function w. The idea behind Multi-resolution STFT Error (MRSE), as proposed in [65], is to average errors from STFT's with different configurations, i.e. FFT size, window size, and frame shift, thereby account for several points in the trade-off.

For a single STFT parameter choice the error between ground truth and generated signals $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$ is defined as

$$L_s(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \mathbb{E}_{\boldsymbol{x}, \tilde{\boldsymbol{x}}}[L_{\text{sc}}(\boldsymbol{x}, \tilde{\boldsymbol{x}}) + L_{\text{mag}}(\boldsymbol{x}, \tilde{\boldsymbol{x}})]. \tag{36}$$

Here L_{sc} and L_{mag} are the *spectral convergence* and *log STFT magnitude* losses respectively, which are defined as

$$L_{\text{sc}}(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \frac{\| \, |\text{STFT}\{\boldsymbol{x}\}| - |\text{STFT}\{\tilde{\boldsymbol{x}}\}| \, \|_F}{\| \, |\text{STFT}\{\boldsymbol{x}\}| \, \|_F}, \tag{37}$$

$$L_{\text{mag}}(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \frac{1}{KM} \| \log |\text{STFT}\{\boldsymbol{x}\}| - \log |\text{STFT}\{\tilde{\boldsymbol{x}}\}| \|_1, \tag{38}$$

[1] For calculations this open-source implementation was used: github.com/gudgud96/frechet-audio-distance

where K and M again denote the number of frequency and time samples. The magnitude of the short term Fourier transform of a signal is denoted as $|\mathrm{STFT}\{\cdot\}|$, $\|\cdot\|_F$ the Frobenius norm, and $\|\cdot\|_1$ the L_1 norm.

For D different STFT parameter settings, the MRSE is then defined as

$$L_{\mathrm{MRSE}}(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \frac{1}{D} \sum_{i=1}^{D} L_s^{(i)}(\boldsymbol{x}, \tilde{\boldsymbol{x}}). \tag{39}$$

4 Data

The LJSpeech dataset [26] is an English language public domain dataset of 13100 audio clips of a single speaker reading from non-fiction books published between 1884 and 1964. In total it contains around 24 hours or 2.6 GB of audio, where clip lengths range between 1 to 10 seconds. The audio is 16-bit single-channel mono with a sample rate of 22050 Hz, and has been widely used by a range of previous text-to-speech and vocoder studies [58, 36, 40].

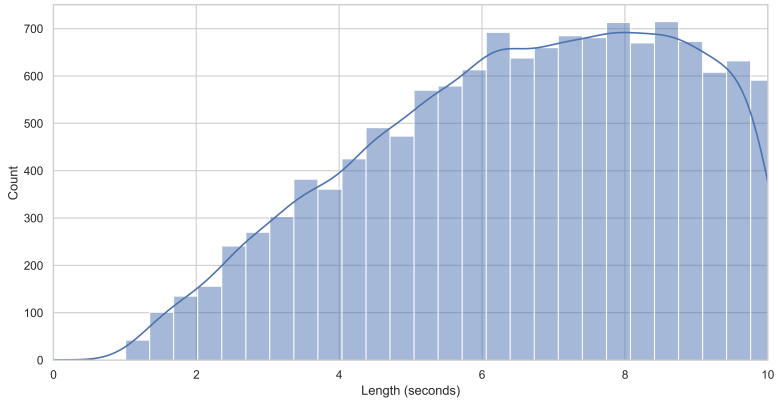

Figure 8: Distribution of audio clip lengths in the LJSpeech dataset [26]. The clips range from 0 to 10 seconds with a majority being over 5 seconds long.

The dataset was chosen because of its popularity and to enable comparison of results to past studies. Compared to multi-speaker datasets [68, 3], LJSpeech provides a reasonable amount of data given limited time and computational resources. Worth noting is that the voice characteristics of a single-speaker TTS model almost entirely follow those of the data it was trained on, such as LJSpeech. In order to obtain a unique, specific voice, which might be desirable in a commercial application, retraining using specific data would be necessary. Alternative approaches such as multi-speaker models [18, 27], and voice cloning [4], also exist, but are out of scope for this thesis.

5 Method

TODO: Introduction

The set of 13 100 audio files was divided into a training, validation, and test set of sizes 13 000, 5, and 95 respectively, as done in [40].

DiffWave was trained using the L1 version of the loss function defined in equation 28 using algorithm 1, as it was reported to produce better results [11]. A single training example was generated as a random audio clip from the training set, from which a mel spectrogram was created. This mel spectrogram was then used to condition the model, along with a time step index, in order to predict the noise added to the training sample. For all experiments a batch size of 16 was used, along with a learning rate of $\eta = 2 \cdot 10^{-4}$. Code was obtained from an open-source implementation [2].

Mel spectrograms were obtained using Librosa 0.9.2 with settings identical to the ones used by HiFi-GAN [35]. These include 1024 FFT components, 80 mel bins, a hop length of 256, a window length of 1024, as well as a minimum and maximum frequency cutoffs of 0 and 8000 Hz. In addition to this a Hann window was used as well as dynamic range compression.

All models were trained for 500k iterations respectively.

How was PriorGrad trained?

Griffin Lim baseline

Evaluation?

Other frameworks used? W&B?

Weighted sampling

GPU details

Comparative experiments? Svårt när de inte är gjorda.

[2] github.com/lmnt-com/diffwave

6 Results

TODO: Section introduction

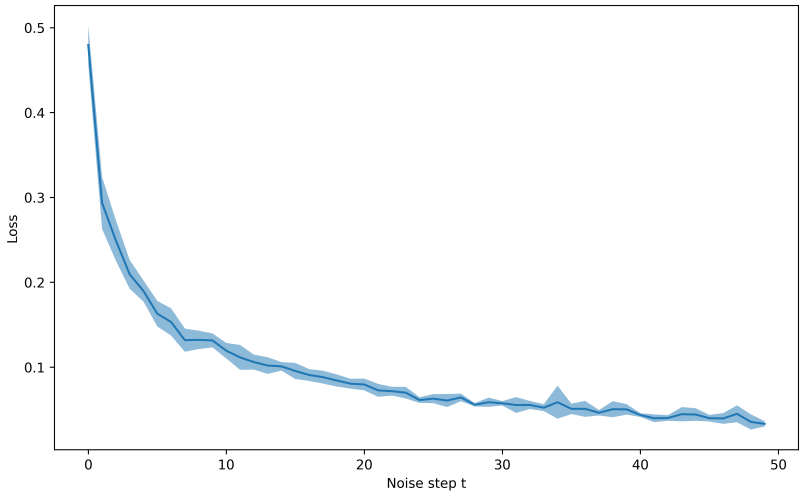

Figure 9: Loss L_{simple} per noise step t averaged for 10 000 training steps of DiffWave, including standard deviation. Note that the average loss is higher for earlier steps in the diffusion process, i.e. when the data is close to the real distribution, as opposed to when it is close to the prior noise.

The Griffin-Lim reconstruction is obtained by inverting the mel spectrogram of each audio file in the test set using the same original STFT settings using 32 iterations.

Model	**LS-MAE**	**FAD**	**PSNR**	**MRSE**	**RTF** (CPU)	**RTF** (GPU)
Griffin-Lim	0.302	3.465	25.517	1.131	0.150	-
DiffWave L1 (500)	-	-	-	-	-	-
PriorGrad L2 (500k)	-	-	-	-	-	-
HiFi-GAN	0.197	0.099	31.256	1.017	0.333	-

Table 1: Vocoder test set results

A sample rate of 22050 Hz was used to generate Mel Spectrograms, where 80 Mel features, an FFT size of 1024, hop size of 256, and a window size of 1024 was used. In addition to this, dynamic range compression was applied to the resulting Mel Spectrograms.

The PSNR was calculated using a custom PyTorch implementation to min-max-normalized Mel Spectrograms.

The MRSE was calculated for three different settings identical to the original paper where it was presented [65], and was done using an open source implementation.[3]

The real-time factor was obtained as the ratio between the Mel Spectrogram to audio-conversion and the length of the audio file. The final RTF was obtained as the mean of all real-time factors obtained from the test set.

[3] github.com/csteinmetz1/auraloss

7 Discussion

Distillation could be used to create a smaller model which id able to achieve similar audio quality to a teacher model.

Only model the residual of an existing Vocoder? Such as HiFi-GAN, FastSpeech 2 (?) or Griffin Lim. (https://arxiv.org/abs/2212.14518)

7.1 Future work

7.2 Conclusion

References

[1] Yannis Agiomyrgiannakis. Vocaine the vocoder and applications in speech synthesis. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4230–4234. IEEE, 2015.

[2] Ehab A AlBadawy, Andrew Gibiansky, Qing He, Jilong Wu, Ming-Ching Chang, and Siwei Lyu. Vocbench: A neural vocoder benchmark for speech synthesis. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 881–885. IEEE, 2022.

[3] Rosana Ardila, Megan Branson, Kelly Davis, Michael Henretty, Michael Kohler, Josh Meyer, Reuben Morais, Lindsay Saunders, Francis M Tyers, and Gregor Weber. Common voice: A massively-multilingual speech corpus. *arXiv preprint arXiv:1912.06670*, 2019.

[4] Sercan Arik, Jitong Chen, Kainan Peng, Wei Ping, and Yanqi Zhou. Neural voice cloning with a few samples. *Advances in neural information processing systems*, 31, 2018.

[5] Sercan Ö Arık, Mike Chrzanowski, Adam Coates, Gregory Diamos, Andrew Gibiansky, Yongguo Kang, Xian Li, John Miller, Andrew Ng, Jonathan Raiman, et al. Deep voice: Real-time neural text-to-speech. In *International Conference on Machine Learning*, pages 195–204. PMLR, 2017.

[6] François Auger, Éric Chassande-Mottin, and Patrick Flandrin. On phase-magnitude relationships in the short-time fourier transform. *IEEE Signal Processing Letters*, 19(5):267–270, 2012.

[7] Anderson R Avila, Hannes Gamper, Chandan Reddy, Ross Cutler, Ivan Tashev, and Johannes Gehrke. Non-intrusive speech quality assessment using neural networks. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 631–635. IEEE, 2019.

[8] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in Neural Information Processing Systems*, 33:12449–12460, 2020.

[9] Mikołaj Bińkowski, Jeff Donahue, Sander Dieleman, Aidan Clark, Erich Elsen, Norman Casagrande, Luis C Cobo, and Karen Simonyan. High fidelity speech synthesis with adversarial networks. *arXiv preprint arXiv:1909.11646*, 2019.

[10] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[11] Nanxin Chen, Yu Zhang, Heiga Zen, Ron J Weiss, Mohammad Norouzi, and William Chan. Wavegrad: Estimating gradients for waveform generation. *arXiv preprint arXiv:2009.00713*, 2020.

[12] Zehua Chen, Xu Tan, Ke Wang, Shifeng Pan, Danilo Mandic, Lei He, and Sheng Zhao. Infergrad: Improving diffusion models for vocoder by considering inference in training. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8432–8436. IEEE, 2022.

[13] Zehua Chen, Yihan Wu, Yichong Leng, Jiawei Chen, Haohe Liu, Xu Tan, Yang Cui, Ke Wang, Lei He, Sheng Zhao, et al. Resgrad: Residual denoising diffusion probabilistic models for text to speech. *arXiv preprint arXiv:2212.14518*, 2022.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

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

[16] Gunnar Fant. *Acoustic theory of speech production*. Number 2. Walter de Gruyter, 1970.

[17] David Foster. *Generative deep learning: teaching machines to paint, write, compose, and play*. O'Reilly Media, 2019.

[18] Andrew Gibiansky, Sercan Arik, Gregory Diamos, John Miller, Kainan Peng, Wei Ping, Jonathan Raiman, and Yanqi Zhou. Deep voice 2: Multi-speaker neural text-to-speech. *Advances in neural information processing systems*, 30, 2017.

[19] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[20] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[21] Daniel Griffin and Jae Lim. Signal estimation from modified short-time fourier transform. *IEEE Transactions on acoustics, speech, and signal processing*, 32(2):236–243, 1984.

[22] Monson Hayes, Jae Lim, and Alan Oppenheim. Signal reconstruction from phase or magnitude. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(6):672–680, 1980.

[23] 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.

[24] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[25] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

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

[27] Ye Jia, Yu Zhang, Ron Weiss, Quan Wang, Jonathan Shen, Fei Ren, Patrick Nguyen, Ruoming Pang, Ignacio Lopez Moreno, Yonghui Wu, et al. Transfer learning from speaker verification to multispeaker text-to-speech synthesis. *Advances in neural information processing systems*, 31, 2018.

[28] Alexia Jolicoeur-Martineau, Ke Li, Rémi Piché-Taillefer, Tal Kachman, and Ioannis Mitliagkas. Gotta go fast when generating data with score-based models. *arXiv preprint arXiv:2105.14080*, 2021.

[29] Nal Kalchbrenner, Erich Elsen, Karen Simonyan, Seb Noury, Norman Casagrande, Edward Lockhart, Florian Stimberg, Aaron Oord, Sander Dieleman, and Koray Kavukcuoglu. Efficient neural audio synthesis. In *International Conference on Machine Learning*, pages 2410–2419. PMLR, 2018.

[30] Hideki Kawahara, Ikuyo Masuda-Katsuse, and Alain De Cheveigne. Restructuring speech representations using a pitch-adaptive time–frequency smoothing and an instantaneous-frequency-based f0 extraction: Possible role of a repetitive structure in sounds. *Speech communication*, 27(3-4):187–207, 1999.

[31] 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.

[32] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[33] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018.

[34] Dennis H Klatt. Software for a cascade/parallel formant synthesizer. *the Journal of the Acoustical Society of America*, 67(3):971–995, 1980.

[35] 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, 2020.

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

[37] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[38] Max WY Lam, Jun Wang, Dan Su, and Dong Yu. Bddm: Bilateral denoising diffusion models for fast and high-quality speech synthesis. *arXiv preprint arXiv:2203.13508*, 2022.

[39] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[40] Sang-gil Lee, Heeseung Kim, Chaehun Shin, Xu Tan, Chang Liu, Qi Meng, Tao Qin, Wei Chen, Sungroh Yoon, and Tie-Yan Liu. Priorgrad: Improving conditional denoising diffusion models with data-driven adaptive prior. *arXiv preprint arXiv:2106.06406*, 2021.

[41] Yulong Lu and Jianfeng Lu. A universal approximation theorem of deep neural networks for expressing probability distributions. *Advances in neural information processing systems*, 33:3094–3105, 2020.

[42] Radford M Neal. Annealed importance sampling. *Statistics and computing*, 11(2):125–139, 2001.

[43] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021.

[44] 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.

[45] Tom Le Paine, Pooya Khorrami, Shiyu Chang, Yang Zhang, Prajit Ramachandran, Mark A Hasegawa-Johnson, and Thomas S Huang. Fast wavenet generation algorithm. *arXiv preprint arXiv:1611.09482*, 2016.

[46] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[47] 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.

[48] Ryan Prenger, Rafael Valle, and Bryan Catanzaro. Waveglow: A flow-based generative network for speech synthesis. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3617–3621. IEEE, 2019.

[49] Dario Rethage, Jordi Pons, and Xavier Serra. A wavenet for speech denoising. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5069–5073. IEEE, 2018.

[50] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.

[51] Antony W Rix, John G Beerends, Michael P Hollier, and Andries P Hekstra. Perceptual evaluation of speech quality (pesq)-a new method for speech quality assessment of telephone networks and codecs. In *2001 IEEE international conference on acoustics, speech, and signal processing. Proceedings (Cat. No. 01CH37221)*, volume 2, pages 749–752. IEEE, 2001.

[52] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[53] 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.

[54] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. (arXiv:1503.03585), Nov 2015. arXiv:1503.03585 [cond-mat, q-bio, stat].

[55] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32, 2019.

[56] Stanley Smith Stevens, John Volkmann, and Edwin Broomell Newman. A scale for the measurement of the psychological magnitude pitch. *The journal of the acoustical society of america*, 8(3):185–190, 1937.

[57] Robert C Streijl, Stefan Winkler, and David S Hands. Mean opinion score (mos) revisited: methods and applications, limitations and alternatives. *Multimedia Systems*, 22(2):213–227, 2016.

[58] Xu Tan, Tao Qin, Frank Soong, and Tie-Yan Liu. A survey on neural speech synthesis. *arXiv preprint arXiv:2106.15561*, 2021.

[59] Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*, 2015.

[60] Tomoki Toda, Alan W Black, and Keiichi Tokuda. Voice conversion based on maximum-likelihood estimation of spectral parameter trajectory. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(8):2222–2235, 2007.

[61] 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.

[62] Wolfgang Von Kempelen, Ritter von Kempelen de Pázmánd, Autriche Mécanicien, Ritter von Kempelen de Pázmánd, Österreich Mechaniker, Ritter von Kempelen de Pázmánd, and Austria Mechanic. *Mechanismus der menschlichen Sprache nebst der Beschreibung seiner sprechenden Maschine*. Bei JV Degen, 1791.

[63] 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.

[64] Daniel Watson, Jonathan Ho, Mohammad Norouzi, and William Chan. Learning to efficiently sample from diffusion probabilistic models. *arXiv preprint arXiv:2106.03802*, 2021.

[65] Ryuichi Yamamoto, Eunwoo Song, and Jae-Min Kim. Parallel wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6199–6203. IEEE, 2020.

[66] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[67] Heiga Ze, Andrew Senior, and Mike Schuster. Statistical parametric speech synthesis using deep neural networks. In *2013 ieee international conference on acoustics, speech and signal processing*, pages 7962–7966. IEEE, 2013.

[68] Heiga Zen, Viet Dang, Rob Clark, Yu Zhang, Ron J Weiss, Ye Jia, Zhifeng Chen, and Yonghui Wu. Libritts: A corpus derived from librispeech for text-to-speech. *arXiv preprint arXiv:1904.02882*, 2019.

[69] Heiga Zen, Keiichi Tokuda, and Alan W Black. Statistical parametric speech synthesis. *speech communication*, 51(11):1039–1064, 2009.

[70] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4490–4499, 2018.