WaveGrad: Estimating Gradients for Waveform Generation

Nanxin Chen[*]
Johns Hopkins University, Center for Language and Speech Processing
bobchennan@jhu.edu

Yu Zhang[†] **Heiga Zen, Ron J. Weiss, Mohammad Norouzi, William Chan**[†]
Google Research, Brain Team
{ngyuzh,heigazen,ronw,mnorouzi,williamchan}@google.com

Abstract

This paper introduces *WaveGrad*, a conditional model for waveform generation through estimating gradients of the data density. This model is built on the prior work on score matching and diffusion probabilistic models. It starts from Gaussian white noise and iteratively refines the signal via a gradient-based sampler conditioned on the mel-spectrogram. WaveGrad is non-autoregressive, and requires only a constant number of generation steps during inference. It can use as few as 6 iterations to generate high fidelity audio samples. WaveGrad is simple to train, and implicitly optimizes for the weighted variational lower-bound of the log-likelihood. Empirical experiments reveal WaveGrad to generate high fidelity audio samples matching a strong likelihood-based autoregressive baseline with less sequential operations.

1 Introduction

Deep generative models have revolutionized speech synthesis (Oord et al., 2016; Sotelo et al., 2017; Wang et al., 2017; Biadsy et al., 2019; Jia et al., 2019). Autoregressive models, in particular, have been popular for raw audio generation due to their tractable likelihoods, simple inference procedures, and high fidelity samples (Oord et al., 2016; Mehri et al., 2016; Kalchbrenner et al., 2018; Song et al., 2019a; Valin & Skoglund, 2019). However, autoregressive models require a large number of sequential computational steps to generate one audio sample, which makes it challenging to deploy these models in real-world production applications, such as digital voice assistants on smart speakers, even using specialized hardware designed for neural networks.

There has been a plethora of research into non-autoregressive models for audio generation, including normalizing flow (NF) models like inverse autoregressive flow (IAF) (Oord et al., 2018; Ping et al., 2019), generative flow (Glow) (Prenger et al., 2019; Kim et al., 2019), and continuous normalizing flow (CNF) (Kim et al., 2020; Wu & Ling, 2020), implicit generative models like generative adversarial network (GAN) (Donahue et al., 2018; Engel et al., 2019; Kumar et al., 2019; Yamamoto et al., 2020; Bikowski et al., 2020; Yang et al., 2020; McCarthy & Ahmed, 2020) and energy score (Gritsenko et al., 2020), variational models like variational auto-encoder (VAE) (Peng et al., 2020), models inspired by digital signal processing (Ai & Ling, 2020; Engel et al., 2020), and those by the speech production mechanism (Juvela et al., 2019; Wang et al., 2020). Although these models improve inference speed thanks to their architectures offering less sequential operations, they often yield lower quality samples than the autoregressive models.

This paper introduces *WaveGrad*, a conditional generative model of waveform that estimates the gradients of the data log-density as opposed to the density itself. WaveGrad offers a natural way to trade inference speed for sample quality by adjusting the number of refinement steps and bridges the gap between non-autoregressive and autoregressive models in terms of audio quality. Figure 1

[*]Work done during an internship at Google Brain.
[†]Equal contribution.

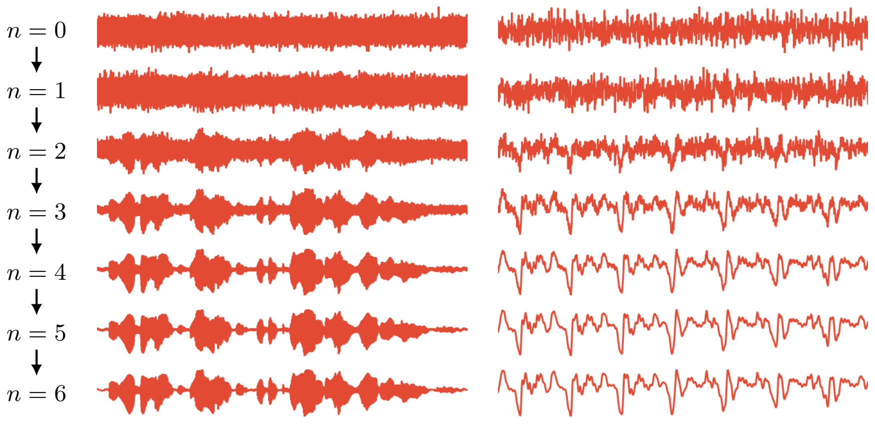

Figure 1: A visualization of the inference process of WaveGrad. Starting from Gaussian noise ($n = 0$), it applies gradient-based sampling with as few as 6 iterations to achieve high fidelity audio ($n = 6$). The left column visualizes the signal after each step of a gradient-based sampler, and the right column zooms in on a 50 ms segment.

visualizes the inference process, where WaveGrad starts from Gaussian noise, then iteratively refines the generated signal using a gradient-based sampler, using a constant number of refinement steps.

WaveGrad builds on a class of generative models that emerges through learning the gradient of the data log-density, also known as the Stein score function (Hyvärinen, 2005; Vincent, 2011). During inference, one can rely on the gradient estimate of the data log-density and use gradient-based samplers (e.g., Langevin dynamics) to sample from the model (Song & Ermon, 2019). Promising results have achieved on image synthesis (Song & Ermon, 2019; 2020) and shape generation (Cai et al., 2020). Closely related are diffusion probabilistic models (Sohl-Dickstein et al., 2015), which capture the output distribution through a Markov chain of latent variables. Although these models do not offer tractable likelihoods, one can optimize a (weighted) variational lower-bound on the log-likelihood. The training objective can be reparameterized to resemble deonising score matching (Vincent, 2011), and can be interpreted as estimating the data log-density gradients. During inference, the model is non-autoregressive requiring only a constant number of generation steps, and follows a Langevin dynamics-like sampler to generate the output beginning from Gaussian noise.

Key contributions of this paper are summarized as follows:

- WaveGrad combines techniques from denoising score matching (Song et al., 2019b; Song & Ermon, 2020) and diffusion probabilistic models (Sohl-Dickstein et al., 2015; Ho et al., 2020) to address conditional speech synthesis.
- We conduct experiments with two variants of the WaveGrad model; (1) WaveGrad conditioned on a discrete refinement step index, (2) WaveGrad conditioned on a continuous scalar indicating the noise level. We find that the continuous variant is more effective, especially because once the model is trained, different number of refinement steps can be used for waveform generation.
- We demonstrate that WaveGrad is capable of generating high fidelity audio samples, comparable with one of the best autoregressive models (Kalchbrenner et al., 2018). It is still capable to generating high fidelity samples using as few as 6 refinement steps.

The rest of this paper is organized as follows. Section 2 describes the proposed WaveGrad model. Section 3 discusses the relationship between the prior work and WaveGrad. Section 4 shows experimental results. Concluding remark is given at Section 5.

2 Estimating Gradients for Waveform Generation

We begin with a brief review of the Stein score function, Langevin dynamics, and score matching. The Stein score function (Hyvärinen, 2005) is the gradient of the data log-density $\log p(y)$ with respect to the datapoint y:

$$s(y) = \nabla_y \log p(y). \tag{1}$$

Given the Stein score function $s(\cdot)$, one can draw samples from the corresponding density, $\tilde{y} \sim p(y)$, via Langevin dynamics, which can be interpreted as stochastic gradient ascent in the data space:

$$\tilde{y}_{i+1} = \tilde{y}_i + \frac{\epsilon}{2} s(\tilde{y}_i) + \sqrt{\epsilon}\, z_i, \tag{2}$$

where $\epsilon > 0$ is the step size, $z_i \sim \mathcal{N}(0, I)$, and I denotes an identity matrix.

One can then imagine building a generative model by training a model to learn the Stein score function directly, and then use Langevin dynamics for inference. This approach, known as score matching (Hyvärinen, 2005; Vincent, 2011), has seen recent success in applications such as image generation (Song & Ermon, 2019; 2020) and shape generation (Cai et al., 2020). The denoising score matching objective (Vincent, 2011) takes the form:

$$\mathbb{E}_{y \sim p(y)}\, \mathbb{E}_{\tilde{y} \sim q(\tilde{y} \mid y)} \left[\left\| s_\theta(\tilde{y}) - \nabla_{\tilde{y}} \log q(\tilde{y} \mid y) \right\|_2^2 \right], \tag{3}$$

where $p(\cdot)$ is the data distribution, and $q(\cdot)$ is a noise distribution.

Recently, Song & Ermon (2019) proposed a weighted denoising score matching objective, in which data is perturbed with different levels of Gaussian noise, and the score function $s_\theta(\tilde{y}, \sigma)$ is conditioned on σ, the standard deviation of the noise used:

$$\sum_{\sigma \in S} \lambda(\sigma)\, \mathbb{E}_{y \sim p(y)}\, \mathbb{E}_{\tilde{y} \sim \mathcal{N}(y, \sigma)} \left[\left\| s_\theta(\tilde{y}, \sigma) - \frac{\tilde{y} - y}{\sigma^2} \right\|_2^2 \right], \tag{4}$$

where S is a set of standard deviation values that one perturbs the data with, and $\lambda(\sigma)$ is a weighting function for different σ. WaveGrad is a variant of this approach applied to learning *conditional* generative models of the form $p(y \mid x)$. WaveGrad learns the gradient of the data density, and uses a sampler similar to Langevin dynamics for inference.

The denoising score matching framework relies on a noise distribution to provide support for learning the gradient of the data log density (i.e., q in Equation 3, and $\mathcal{N}(\cdot, \sigma)$ in Equation 4). The choice of the noise distribution is critical for achieving high quality samples (Song & Ermon, 2020). WaveGrad relies on the diffusion model framework (Sohl-Dickstein et al., 2015; Ho et al., 2020) to generate the noise distribution used to learn the score function.

2.1 WaveGrad as a Diffusion Probabilistic Model

Ho et al. (2020) made the observation that diffusion probabilistic models (Sohl-Dickstein et al., 2015) and score matching objectives (Song & Ermon, 2019; Vincent, 2011; Song & Ermon, 2020) are closely related. As such, we will first introduce WaveGrad as a diffusion probabilistic model (Sohl-Dickstein et al., 2015).

We adapt the diffusion model setup from Ho et al. (2020), from unconditional image generation to conditional raw audio waveform generation. WaveGrad models the conditional distribution $p_\theta(y_0 \mid x)$ where y_0 is the waveform and x is the conditioning features corresponding to y_0, such as linguistic features derived from the corresponding text, ground-truth mel-spectrogram extracted from y_0, or acoustic features predicted by a Tacotron-style text-to-speech synthesis model (Shen et al., 2018)):

$$p_\theta(y_0 \mid x) := \int p_\theta(y_{0:N} \mid x)\, \mathrm{d}y_{1:N}, \tag{5}$$

where $y_1, \ldots, y_N$ is a series of latent variables, each of which are of the same dimension as the data y_0, and N is the number of latent variables (iterations). The generative distribution $p_\theta(y_{0:N} \mid x)$ is called the denoising process (or reverse process), and is defined through the Markov chain:

$$p_\theta(y_{0:N} \mid x) := p(y_N) \prod_{n=1}^{N} p_\theta(y_{n-1} \mid y_n, x), \tag{6}$$

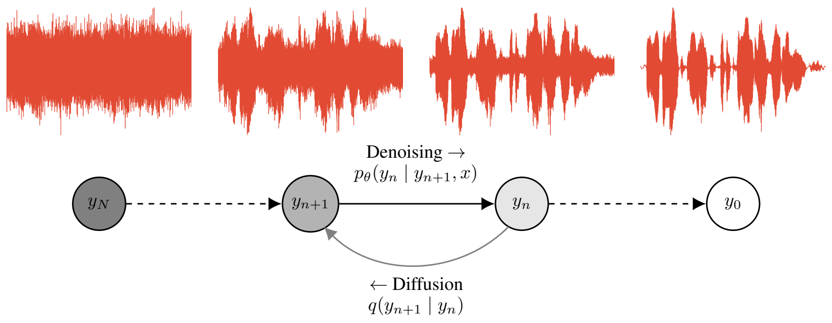

Figure 2: A directed graphical model of the diffusion probabilistic model, adapted from Ho et al. (2020). The diffusion process $q(y_{n+1} \mid y_n)$ iteratively adds Gaussian noise to the signal starting from the waveform y_0; while the denoising process $p_\theta(y_n \mid y_{n+1}, x)$ iteratively removes noise from the signal starting from Gaussian noise y_N.

where each iteration is modelled as a Gaussian transition:

$$p_\theta(y_{n-1} \mid y_n, x) \coloneqq \mathcal{N}\left(y_{n-1}; \mu_\theta(y_n, n, x), \Sigma_\theta(y_n, n, x)\right), \tag{7}$$

starting from Gaussian white noise $p(y_N) = \mathcal{N}(y_N; 0, I)$. The approximate posterior $q(y_{1:N} \mid y_0)$ is called the diffusion process (or forward process), and is defined through the Markov chain:

$$q(y_{1:N} \mid y_0) \coloneqq \prod_{n=1}^{N} q(y_n \mid y_{n-1}), \tag{8}$$

where each iteration adds Gaussian noise:

$$q(y_n \mid y_{n-1}) \coloneqq \mathcal{N}\left(y_n; \sqrt{(1-\beta_n)}\, y_{n-1}, \beta_n I\right), \tag{9}$$

under some (fixed constant) noise schedule $\beta_1, \ldots, \beta_N$. We emphasize the property observed by Ho et al. (2020), the diffusion process can be computed for any step n in a closed form:

$$q(y_n \mid y_0) = \mathcal{N}\left(y_n; \sqrt{\bar{\alpha}_n}\, y_0, (1-\bar{\alpha}_n) I\right), \tag{10}$$

where $\alpha_n \coloneqq 1 - \beta_n$ and $\bar{\alpha}_n \coloneqq \prod_{s=1}^{n} \alpha_s$. During training, we can optimize for the variational lower-bound on the log-likelihood (upper-bound on the negative log-likelihood):

$$-\log p_\theta(y_0 \mid x) \le \mathbb{E}_q\left[-\log \frac{p_\theta(y_0 \mid x)}{q(y_{1:N} \mid y_0)}\right] \tag{11}$$

$$= \mathbb{E}_q\left[-\log p(y_N) - \sum_{n=1}^{N} \log \frac{p_\theta(y_{n-1} \mid y_n, x)}{q(y_n \mid y_{n-1})}\right]. \tag{12}$$

Following Equation 12, it would be natural to parameterize a neural network to model the mean μ_θ and variance Σ_θ of the Gaussian distribution in Equation 7, which would make it possible to directly optimize the KL-divergence with Monte Carlo estimates. However, Ho et al. (2020) found it beneficial to simply set Σ_θ to a constant following the β_n schedule, and to reparameterize the neural network to model ϵ_θ, predicting the noise $\epsilon \sim \mathcal{N}(0, I)$ instead of μ_θ. Under this reparameterization, the loss can be written as:

$$\mathbb{E}_{n,\epsilon}\left[C \left\| \epsilon - \epsilon_\theta\left(\sqrt{\bar{\alpha}_n}\, y_0 + \sqrt{1-\bar{\alpha}_n}\, \epsilon, x, n\right)\right\|_2^2\right], \tag{13}$$

where

$$C = \frac{\beta_n^2}{2\sigma_n^2 \alpha_n (1-\bar{\alpha}_n)}. \tag{14}$$

In practice Ho et al. (2020) also found it beneficial to simply drop the C term, and thus leading to a weighted variational lower bound of the log-likelihood. As noted by Ho et al. (2020), ϵ_θ can be interpreted as the score function or the gradient of the data density, and thus the objective resembles score matching as in Equation 4 (Song & Ermon, 2019). Additionally under the parameterization of (Ho et al., 2020), ϵ_θ conditions on the discrete index n, we will discuss this further in the text. We also found that substituting the original L_2 distance metric with L_1 offers better training stability. Figure 2 visualizes the diffusion and denoising processes.

Algorithm 1 Training algorithm for WaveGrad. WaveGrad directly conditions on the continuous noise $\sqrt{\bar{\alpha}}$, and l is from a predefined noise schedule.

```
1: repeat
2:     $y_0 \sim q(y_0)$
3:     $s \sim \mathrm{Uniform}(\{1, \ldots, S\})$
4:     $\sqrt{\bar{\alpha}} \sim \mathrm{Uniform}(l_{s-1}, l_s)$
5:     $\epsilon \sim \mathcal{N}(0, I)$
6:     Take gradient descent step on
           $\nabla_\theta \left\| \epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}}\, y_0 + \sqrt{1 - \bar{\alpha}}\, \epsilon, x, \sqrt{\bar{\alpha}}) \right\|_1$
7: until converged
```

Algorithm 2 Sampling algorithm for WaveGrad. WaveGrad follows a gradient-based sampler similar to Langevin dynamics to generate samples (Ho et al., 2020).

```
1: $y_N \sim \mathcal{N}(0, I)$
2: for $n = N, \ldots, 1$ do
3:     $z \sim \mathcal{N}(0, I)$ if $n > 1$, else $z = 0$
4:     $y_{n-1} = \frac{1}{\sqrt{\alpha_n}} \left( y_n - \frac{1-\alpha_n}{\sqrt{1-\bar{\alpha}_n}} \epsilon_\theta(y_n, x, \sqrt{\bar{\alpha}_n}) \right) + \sigma_n z$
5: end for
6: return $y_0$
```

2.2 Noise Schedule and Conditioning on Noise Level

In the score matching setup, Song & Ermon (2019; 2020) noted the importance of the noise distribution during training. The noise distribution is important since it provides support for model gradient distribution. The diffusion framework can be viewed as a specific way to provide support to score matching. In the diffusion framework, the noise schedule is parameterized by $\beta_1, \ldots, \beta_N$, as described in the previous section. This is typically determined via some hyperparameter heuristic, e.g., a linear decay schedule (Ho et al., 2020). We found the choice of the noise schedule to be critical towards achieving high fidelity audio in our experiments, especially when trying to minimize the number of inference iterations N to make inference efficient. A schedule with superfluous noise may result in a model unable to recover the high frequency detail of the waveform, while a schedule with too little noise may result in a model that converges poorly during inference. Song & Ermon (2020) provides some insights about how to tune the noise schedule under the framework of score matching. We will connect some of these insights and apply them to WaveGrad under the diffusion framework.

Another closely related problem is determining the hyperparameter N, the number of diffusion/denoising steps. A large N would equip the model with more computational capacity, and may improve sample quality. However using a small N would result in faster inference and lower computational costs. Song & Ermon (2019) adopted $N = 10$ to generate 32×32 images, while Ho et al. (2020) used 1,000 iterations to generate high resolution 256×256 images. In our case, WaveGrad generates audio sampled at 24 kHz, containing 24,000 samples per second.

We found that tuning both the noise schedule and N in conjunction with each other was critical to attaining high fidelity audio, especially when N is small. If these hyperparameters are poorly tuned, the training sampling procedure may provide deficient support for the distribution. Consequently, during inference, our sampler may converge poorly when the sampling trajectory encounters regions that deviate from the conditions seen during training. However, tuning these hyperparameters can be costly due to the large search space, as a large number of models needed to be trained and evaluated. We make empirical observations and discuss this in more details in Section 4.4.

We address some of the issues above in our WaveGrad implementation. First, compared to the diffusion probabilistic model from Ho et al. (2020), we reparameterize the model from conditioning on the discrete iteration index n to conditioning on the continuous noise level $\bar{\alpha}$. This reparameterization allows us (1) not to rely on discrete indices; (2) to enable the model to directly condition on the

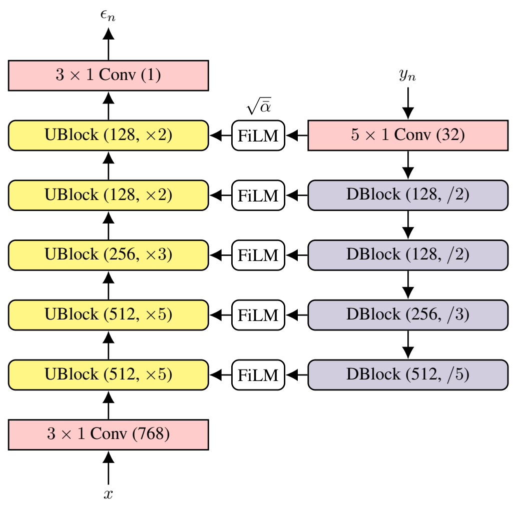

Figure 3: WaveGrad network architecture. The inputs consists of the mel-spectrogram conditioning signal x, the noisy waveform generated from the previous iteration y_n, and the noise level $\sqrt{\bar{\alpha}}$. The model produces ϵ_n at each iteration, which can be interpreted as the direction to update y_n.

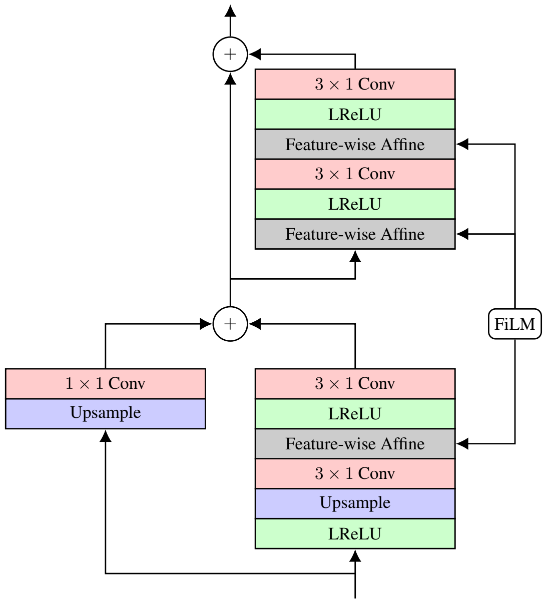

Figure 4: A block diagram of the Upsampling Block (UBlock). We upsample the signal modulated with information from the FiLM module.

amount of noise. The loss becomes

$$\mathbb{E}_{\bar{\alpha},\epsilon}\left[\left\|\epsilon - \epsilon_\theta\left(\sqrt{\bar{\alpha}_n}\,y_0 + \sqrt{1-\bar{\alpha}_n}\,\epsilon, x, \sqrt{\bar{\alpha}}\right)\right\|_1\right], \tag{15}$$

A similar approach was also used in the score matching framework (Song & Ermon, 2019; 2020), wherein they conditioned on the noise variance.

There is one minor technical issue we must resolve in this approach. In the diffusion probabilistic model training procedure conditioned on the discrete iteration index (Equation 13), we would sample $n \sim \text{Uniform}(\{1, \ldots, N\})$, and then proceed to compute its corresponding α_n. In the case of the directly conditioning on the continuous noise, we need to determine a sampling procedure that directly samples $\bar{\alpha}$. Recall that $\bar{\alpha}_n := \prod_s^n (1-\beta_s) \in [0,1]$, while we could simply just sample from the uniform distribution $\bar{\alpha} \sim \text{Uniform}(0,1)$, however, we found this to provide poor empirical results. We instead use a simple hierarchical sampling method that mimics the discrete sampling strategy. We first define a noise schedule with S iterations and compute all its $\sqrt{\bar{\alpha}_s}$:

$$l_0 = 1, \qquad l_s = \sqrt{\prod_{i=1}^{s}(1-\beta_s)}. \tag{16}$$

We first sample a segment $s \sim U(\{1, \ldots, S\})$, which provides a segment (l_{s-1}, l_s), and then sample from this segment uniformly to compute $\sqrt{\bar{\alpha}}$. The full WaveGrad training algorithm with our sampling procedure is illustrated in Algorithm 1.

One benefit of the the WaveGrad model is that it needs to be trained only once, yet inference can be run over a large space of trajectories without the need to be retrained. To be specific, once we train a model, we can use arbitrary different number of N iterations during inference, making it possible to explicitly trade off between inference computation and output quality using the same model. This also makes fast hyperparameter search possible, as we will illustrate in Section 4.4. The full WaveGrad inference algorithm is explained in Algorithm 2 which is a gradient-based sampler akin to Langevin dynamics.

2.3 Neural Network Architecture

We will now describe the neural network architecture of WaveGrad, the full architecture is visualized in Figure 3. To convert the mel-spectrogram signal (80 Hz) into raw audio (24 kHz), five upsampling blocks (UBlock) are applied to gradually upsample the temporal dimension by factors of 5, 5, 3, 2, 2,

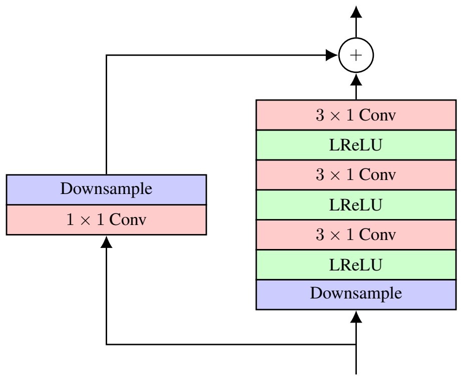

Figure 5: A block diagram of the downsampling block (DBlock).

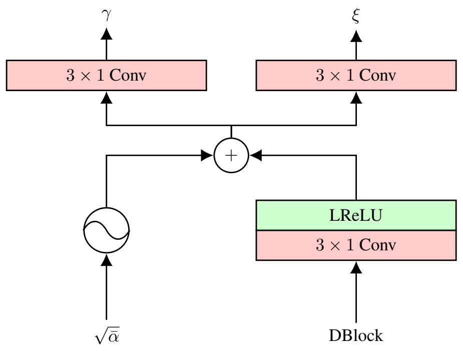

Figure 6: A block diagram of feature-wise linear modulation (FiLM) module. We condition on the noise level $\sqrt{\bar{\alpha}}$ of diffusion/denoising process, and pass it to a positional encoding function.

with the number of channels of 512, 512, 256, 128, 128 respectively. Additionally, one convolutional layer is added before and after these blocks.

The UBlock is illustrated in Figure 4. Each UBlock includes two residual blocks (He et al., 2016). Neural audio generation models often use large receptive field (Oord et al., 2016; Bikowski et al., 2020; Yamamoto et al., 2020). Dilation factors of four convolutional layers are 1, 2, 1, 2 for the first two UBlocks and 1, 2, 4, 8 for the rest. For the large model, we use 1, 2, 4, 8 for all UBlocks.

As an iterative approach, the network prediction is also conditioned on noisy waveform $\sqrt{\bar{\alpha}_n} y_0 + \sqrt{1-\bar{\alpha}_n}\epsilon$. Contrary to UBlock, downsampling blocks (DBlock) are introduced to downsample the temporal dimension of noisy waveform. DBlock is illustrated in Figure 5 and it is similar to UBlock except that only one residual block is included. The dilation factors are 1, 2, 4 in the main branch. Orthogonal initialization (Saxe et al., 2014) is used for all UBlocks and DBlocks.

The Feature-wise linear modulation (FiLM) (Dumoulin et al., 2018) module combines information from both noisy waveform and input mel-spectrogram. We also added iteration index n which indicates the noise level of the input waveform by using the Transformer sinusoidal positional embedding (Vaswani et al., 2017). To condition on the noise level directly, n is replaced by $\sqrt{\bar{\alpha}}$ and a linear scale $C = 5000$ is applied. The FiLM module produces both scale and bias vectors given inputs, which are used in a UBlock for feature-wise affine transformation as

$$\gamma(D, \sqrt{\bar{\alpha}}) \odot U + \xi(D, \sqrt{\bar{\alpha}}), \tag{17}$$

where γ and ξ correspond to the scaling and shift vectors from the FiLM module, D is the output from corresponding DBlock, U is an intermediate output in the UBlock, and $\odot$ denotes the Hadamard product.

An overview of the FiLM module is illustrated in Figure 6. The structure is inspired by spatially-adaptive denormalization (Park et al., 2019). However batch normalization (Ioffe & Szegedy, 2015) is not applied in our work since each minibatch contains samples with different levels of noise. Batch statistics are not accurate since they are heavily depending on sampled n. Experiment results also verified our assumptions that model trained with batch normalization generates low-quality speech.

3 Related Work

Our work is closely related to and heavily inspired by Sohl-Dickstein et al. (2015), where they applied diffusion probabilistic models to unconditional image synthesis while we apply the diffusion probabilistic models to conditional speech synthesis. This objective also resembles the Noise Conditional Score Networks (NCSN) objective of Song & Ermon (2019). Our work is also similiar to Song & Ermon (2019; 2020), wherein our models condition on the noise level. Denoising score matching (Vincent, 2011) and sliced score matching Song et al. (2019b) also hold similar score matching objectives, however in their work they do not condition on the noise level. Finally, Cai et al. (2020) applied NCSN to model conditional distributions for shape generation, while our focus is waveform generation.

Our work also closely relates to masked-based generative models (Devlin et al., 2019; Lee et al., 2018; Ghazvininejad et al., 2019; Chan et al., 2020; Saharia et al., 2020), insertion-based generative models (Stern et al., 2019; Chan et al., 2019b;a;c; Li & Chan, 2019) and edit-based generative models (Sabour et al., 2019; Gu et al., 2019; Ruis et al., 2019) found in the semi-autoregressive sequence generation literature. In these work, they model discrete tokens as opposed to continuous outputs. In our work, we model the (continuous) gradients on the continuous data, while in their work they model (discrete) edits (e.g., insertion or deletion) on the discrete data. The edits are directions to modify the discrete sequence to regions of higher log-likelihood, this is analogous to our (continuous) gradient. These models can also iteratively refine the outputs during inference (Lee et al., 2018; Ghazvininejad et al., 2019; Chan et al., 2020), while they do not rely on a (continuous) gradient-based sampler, they rely on a (discrete) edit-based sampler. Their work also relies heavily on a noise distribution, for example Bernoulli (Devlin et al., 2019), uniform (Saharia et al., 2020), or even hand-crafted (Chan et al., 2020) to provide support for learning the edit distribution, while in our work we rely on a Markov chain from the diffusion framework (Ho et al., 2020).

Finally, our neural network architecture is heavily inspired by GAN-TTS (Bikowski et al., 2020). Our UBlock is inspired by the GAN-TTS GBlock generator, with the minor difference that we do not use BatchNorm.

4 Experiments

In this section we compare WaveGrad with other neural vocoder and carry out ablations with different noise schedules. We report that the WaveGrad model achieve the same quality to the state-of-the-art autoregressive model, WaveRNN, and being significantly more computationally efficient.

4.1 Model and Training setup

We used Google's proprietary speech dataset consisted of 385 hours of high quality English speech from 84 professional voice talents for training models. We chose a female speaker in the training dataset, who had also been used in (Oord et al., 2016; Kalchbrenner et al., 2018; Shen et al., 2018; Bikowski et al., 2020), for evaluation. All speech signals were downsampled to 24 kHz then 128-dimensional mel-spectrogram was computed (window length: 50 ms, frame shift: 12.5 ms, FFT length: 2048, window function: Hanning, lower frequency cut-off: 20 Hz, and upper frequency cut-off: 12 kHz). During training, ground truth mel-spectrogram was used as the conditioning signal for the WaveGrad model, whereas we used the predicted mel-spectrograms from a Tacotron 2 model (Shen et al., 2018) as the conditioning signal during inference. Unlike Shen et al. (2018), preliminary experimental indicated that using ground truth mel-spectrogram as conditioning during training had no regression compared to using predicted mel-spectrogram. This property is beneficial as it can simplify the training process of text-to-speech models: the WaveGrad vocoder model can be trained separately on a large corpus without relying on a pretrained text-to-spectrogram model.

Model Size: Two network size variations were explored: Base and Large. The WaveGrad Base model took 24 frames corresponding to 0.3 second audio as input during training. Our network is fully convolutional, non-autoregressive and highly parallelizable. We set the batch size to 256 and models were trained on Google Tensor Processing Unit (TPU) v2 Pods for 12 hours. The WaveGrad Base model contained 15 million parameters.

For the WaveGrad Large model, we repeated each UBlock/DBlock twice, one with upsampling/downsample and another without. Each training sample included 60 frames corresponding to a 0.75 second audio segment. We kept the same batch size and trained the model on TPU v3 Pods. The WaveGrad Large model contained 23 million parameters.

Noise Schedule: For the WaveGrad Base model, we tested different noise schedules during training. For 1000 and 50 iterations, we set the forward process variances to constants increasing linearly from β_1 to β_N, defined as Linear(β_1, β_N, N). We used Linear(1×10^{-4}, 0.005, 1000) for 1000 iterations and Linear(1×10^{-4}, 0.05, 50) for 50 iterations. For 25 iteration, a different Fibonacci-

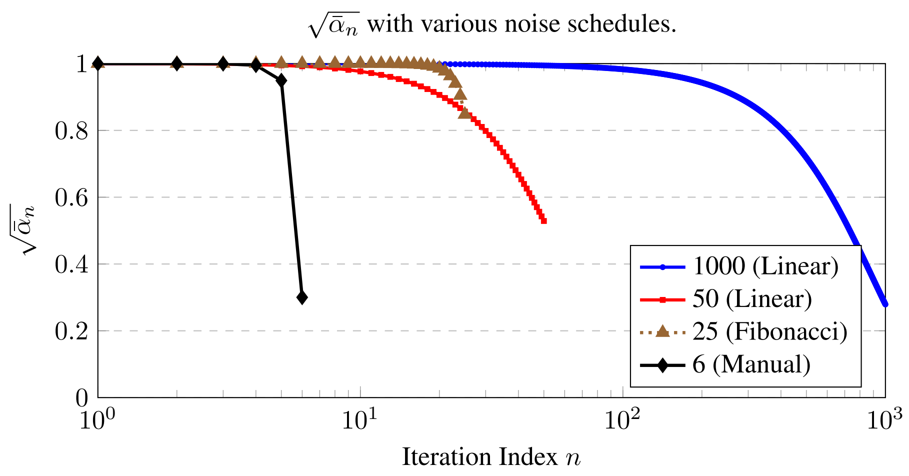

Figure 7: A plot of different noise schedules.

based schedule was adopted (referred to as Fibonacci(N)):

$$\begin{aligned} \beta_0 &= 1 \times 10^{-6} \\ \beta_1 &= 2 \times 10^{-6} \\ \beta_n &= \beta_{n-1} + \beta_{n-2} \quad \forall n \geq 2. \end{aligned} \tag{18}$$

When a fixed schedule was used during training, the same schedule was used during inference. We found a mismatch in the noise schedule degraded the performance. Different noise schedules and corresponding $\sqrt{\bar{\alpha}}$ are plotted in Figure 7.

To sample the noise level $\sqrt{\bar{\alpha}}$, we set the maximal iteration S to 1000 and l_1 to l_S are precomputed from Linear(1×10^{-6}, 0.01, 1000). Unlike the base fixed schedule, WaveGrad support using a different schedule during inference thus "Manual" schedule was also explored. More detailed ablation studies at Section 4.4.

4.2 Evaluation

We used a sample-by-sample autoregressive WaveRNN (Kalchbrenner et al., 2018) conditioned on mel-spectrogram (predicted by a Tacotron 2 model in a teacher-forcing mode) as a baseline. The model had a single long short-term memory (LSTM) layer with 1,024 hidden units, 5 convolutional layers with 512 channels as conditional stack for the mel-spectrogram, and used a 10-component mixture of logistic distributions (MoL) (Salimans et al., 2017) as its output layer to generate 16-bit samples at 24 kHz. It was trained using the same training data as the WaveGrad models. This WaveRNN model had 18 million parameters. Preliminary experiments indicated that further reduction of the hidden units in the LSTM layer would hurt the performance.

To verify the effectiveness of the proposed WaveGrad model, we report subjective listening test results in 5-scale Mean Opinion Score (MOS) in naturalness. The test set included 1,000 sentences. We predicted mel-spectrogram for the test set by the Tacotron 2 model then synthesized audio signals by the baseline WaveRNN and proposed WaveGrad models. Note that the Tacotron 2 model was identical to that used for predicting the mel-spectrogram for conditioning the baseline WaveRNN model. Subjects were asked to rate the naturalness of each stimuli after listening to it. Following previous studies, a five-point Likert scale score (1: Bad, 2: Poor, 3: Fair, 4: Good, 5: Excellent) was adopted and the increment was 0.5. Each subject could evaluate up to six stimuli. Test stimuli were randomly chosen and presented for each subject. Each stimulus was presented to a subject in isolation and was evaluated by one subject. The subjects were paid and native speakers of English living in United States. They were requested to use headphones in a quiet room.

Table 1: Mean opinion scores (MOS) of various models and their confidence intervals. "Ling." and "Mel." in the "Input" column indicates that linguistic features and mel-spectrogram were used as conditioning, respectively. Although different papers listed in "Prior Work" in this table used the same female speaker, their results are not directly comparable due to differences in the training dataset and input representation. However all experiments in "Our Work" in this table are comparable.

Model	Input	Sampling Rate (kHz)	**MOS**
Prior Work			
Autoregressive			
WaveNet (Oord et al., 2016)	Ling.	16	4.21 ± 0.08
WaveNet (Oord et al., 2018)	Ling.	24	4.41 ± 0.07
WaveNet (Shen et al., 2018)	Mel.	24	4.53 ± 0.07
WaveRNN (Kalchbrenner et al., 2018)	Ling.	24	4.46 ± 0.07
Non-autoregressive			
Parallel WaveNet (Oord et al., 2018)	Ling.	24	4.41 ± 0.08
GAN-TTS (Bikowski et al., 2020)	Ling.	24	4.21 ± 0.05
GED (Gritsenko et al., 2020)	Ling.	24	4.25 ± 0.06
Our Work			
Autoregressive			
WaveRNN	Mel.	24	4.49 ± 0.04
Non-autoregressive			
WaveGrad Base (1000 iterations, discrete indices)	Mel.	24	4.47 ± 0.04
WaveGrad Large (1000 iterations, discrete indices)	Mel.	24	4.51 ± 0.04
WaveGrad Base (6 iterations, continuous noise levels)	Mel.	24	4.41 ± 0.03
Ground Truth	–	24	4.58 ± 0.05

4.3 Results

Subjective evaluation results are summarized in Table 1. The models conditioned on discrete indices follow the formulation in Section 2.1, and the models conditioned on continuous noise level follow the formulation in Section 2.2. Our WaveGrad models matches the performance of the autoregressive WaveRNN baseline. Although increasing the model size slightly improved the naturalness, it was not statistically significant. Our WaveGrad Base model with 6 iterations achieves a real time factor (RTF) of 0.2 on an NVIDIA V100 GPU, while still achieving a MOS higher than 4.4. More detailed discussion is in Section 4.4.

4.4 Discussion

To understand the impact of different noise schedules and to reduce the number of iterations in the noise schedule (from 1,000), we explored different noise schedules with fewer iterations. We found that a well-behaved inference schedule should satisfy two conditions:

1. The KL-divergence $D_{\text{KL}}\left(q(y_N \mid y_0) \parallel \mathcal{N}(0, I)\right)$ between y_N and standard normal distribution $\mathcal{N}(0, I)$ needs to be small. Large KL-divergence introduces mismatches between training and inference. To make the KL-divergence small, some βs need to be large enough.
2. β should start with small values. This provides the model training with fine granularity details, which we found crucial for reducing background static noise.

In this subsection, all the experiments were conducted with the WaveGrad Base model. Both subjective and quantitative evaluation results are reported. The quantitative evaluation results include following:

1. Log-mel spectrogram mean squared error metrics (LS-MSE), computed using 50 ms window length and 6.25 ms frame shift;

Table 2: Quantitative and subjective results of different noise schedules used during training and inference, these WaveGrad models were conditioned on the discrete index. The WaveGrad models with 25 and 50 iterations achieved faster than real-time generation (RTF<1) on an NVIDIA V100 GPU.

WaveGrad conditioned on a discrete iteration index

Iterations (schedule)	**LS-MSE (↓)**	**MCD (↓)**	**FFE (↓)**	**MOS (↑)**
25 (Fibonacci)	283	3.93	3.3%	3.86 ± 0.05
50 (Linear $(1 \times 10^{-4}, 0.05)$)	181	3.13	3.1%	4.42 ± 0.04
1000 (Linear $(1 \times 10^{-4}, 0.005)$)	116	2.85	3.2%	4.47 ± 0.04

Table 3: Quantitative and subjective evaluation results of the WaveGrad model conditioned on the noise level (as opposed to the discrete index in Table 2). We trained one model and evaluated it via different inference schemes. We found WaveGrad conditioned on the noise level to yield high fidelity samples even with only a 6 iterations.

WaveGrad conditioned on continuous noise level

Iterations (schedule)	**LS-MSE (↓)**	**MCD (↓)**	**FFE (↓)**	**MOS (↑)**
6 (Manual)	217	3.38	2.8%	4.41 ± 0.04
25 (Fibonacci)	185	3.33	2.8%	4.44 ± 0.04
50 (Linear)	177	3.23	2.7%	4.43 ± 0.04
1000 (Linear)	106	2.85	3.0%	4.46 ± 0.03

2. Mel cepstral distance (MCD) (Kubichek, 1993), a similar MSE metric computed using 13-dimensional mel frequency cepstral coefficient (MFCC) features;
3. F_0 Frame Error (FFE) metric (Chu & Alwan, 2009) which combines Gross Pitch Error (GPE) and Voicing Decision Error (VDE).

For all quantitative evaluations, since ground truth waveform was required, we report results on the validation set which included 50 utterances with ground truth mel-spectrograms. These 50 utterances included audio samples from multiple speakers. We experiment with different noise schedules and number of iterations. These models were trained with conditioning on the discrete index, and the subjective and quantitative evaluation results are described in Table 2.

We also perform a detailed study on the the WaveGrad model conditioned on the continuous noise level in Table 3. Compared to the WaveGrad conditioned on the discrete index and using a fixed training schedule in Table 2, the WaveGrad conditioned on the continuous noise levels generalize better especially when the number of iterations is small. To demonstrate the possibilities with WaveGrad, a 6-iteration inference schedule was explored by sweeping the βs over following possibilities:

$$\begin{aligned} &\{1, 2, 3, 4, 5, 6, 7, 8, 9\} \times \\ &\quad 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1} \end{aligned} \tag{19}$$

Again, we did not need to train individual models for such hyper-parameter tuning. Here we used LS-MSE as a metric for tuning and the result is also reported in the first line of Table 3. Performance degradation was not noticeable. The WaveGrad Base model with 6 iterations achieved real time factor (RTF) = 0.2 on an NVIDIA V100 GPU and RTF = 1.5 on an Intel Xeon CPU (16 cores, 2.3GHz). As we did not optimize the inference code, further speed up can be possible.

5 Conclusion

In this paper, we presented WaveGrad, a conditional model for waveform generation. WaveGrad estimates the gradients of the data density, leveraging on the diffusion probabilistic model (Ho et al., 2020) and score matching framework (Song et al., 2019b; Song & Ermon, 2020). WaveGrad starts from Gaussian white noise and iteratively updates the signal via a gradient-based sampler conditioned on the mel-spectrogram. WaveGrad is non-autoregressive, and requires only a constant number of generation steps during inference. WaveGrad can use as few as 6 iterations to generate high fidelity audio samples. WaveGrad is simple to train, and implicitly optimizes for the weighted variational lower-bound of the log-likelihood. The empirical experiments demonstrated WaveGrad to generate high fidelity audio samples matching a strong autoregressive baseline.

Author Contributions

Nanxin Chen wrote code, proposed the idea, ran all experiments and wrote the paper. Yu Zhang recruited collaborators, co-managed/advised the project, conducted evaluation, debugging model and editing paper. Heiga Zen helped conducting text-to-speech experiments and advised the project. Ron Weiss implemented the objective evaluation metrics and advised the project. Mohammad Norouzi suggested the use of denoising diffusion models for audio generation, and helped advising the project. William Chan conceived the project, wrote code, wrote paper, and co-managed/advised the project.

Acknowledgments

The authors would like to thank Durk Kingma, Yang Song, Kevin Swersky and Yonghui Wu for providing insightful research discussions and feedback.

References

Yang Ai and Zhen-Hua Ling. Knowledge-and-Data-Driven Amplitude Spectrum Prediction for Hierarchical Neural Vocoders. *arXiv preprint arXiv:2004.07832*, 2020.

Fadi Biadsy, Ron J. Weiss, Pedro J. Moreno, Dimitri Kanevsky, and Ye Jia. Parrotron: An End-to-End Speech-to-Speech Conversion Model and its Applications to Hearing-Impaired Speech and Speech Separation. In *INTERSPEECH*, 2019.

Mikoaj Bikowski, Jeff Donahue, Sander Dieleman, Aidan Clark, Erich Elsen, Norman Casagrande, Luis C. Cobo, and Karen Simonyan. High Fidelity Speech Synthesis with Adversarial Networks. In *ICLR*, 2020.

Ruojin Cai, Guandao Yang, Hadar Averbuch-Elor, Zekun Hao, Serge Belongie, Noah Snavely, and Bharath Hariharan. Learning Gradient Fields for Shape Generation. In *ECCV*, 2020.

Harris Chan, Jamie Kiros, and William Chan. Multilingual KERMIT: Its Not Easy Being Generative. In *NeurIPS: Workshop on Perception as Generative Reasoning*, 2019a.

William Chan, Nikita Kitaev, Kelvin Guu, Mitchell Stern, and Jakob Uszkoreit. KERMIT: Generative Insertion-Based Modeling for Sequences. *arXiv preprint arXiv:1906.01604*, 2019b.

William Chan, Mitchell Stern, Jamie Kiros, and Jakob Uszkoreit. An Empirical Study of Generation Order for Machine Translation. *arXiv preprint arXiv:1910.13437*, 2019c.

William Chan, Chitwan Saharia, Geoffrey Hinton, Mohammad Norouzi, and Navdeep Jaitly. Imputer: Sequence Modelling via Imputation and Dynamic Programming. In *ICML*, 2020.

Wei Chu and Abeer Alwan. Reducing F0 Frame Error of F0 Tracking Algorithms under Noisy Conditions with an Unvoiced/Voiced Classification Frontend. In *ICASSP*, 2009.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*, 2019.

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

Vincent Dumoulin, Ethan Perez, Nathan Schucher, Florian Strub, Harm de Vries, Aaron Courville, and Yoshua Bengio. Feature-wise transformations. *Distill*, 2018. doi: 10.23915/distill.00011. https://distill.pub/2018/feature-wise-transformations.

Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts. GANSynth: Adversarial Neural Audio Synthesis. In *ICLR*, 2019.

Jesse Engel, Lamtharn Hantrakul, Chenjie Gu, and Adam Roberts. DDSP: Differentiable Digital Signal Processing. In *ICLR*, 2020.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-Predict: Parallel Decoding of Conditional Masked Language Models. In *EMNLP*, 2019.

Alexey A. Gritsenko, Tim Salimans, Rianne van den Berg, Jasper Snoek, and Nal Kalchbrenner. A Spectral Energy Distance for Parallel Speech Synthesis. *arXiv preprint arXiv:2008.01160*, 2020.

Jiatao Gu, Changhan Wang, and Jake Zhao. Levenshtein Transformer. In *NeurIPS*, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. *arXiv preprint arXiv:2006.11239*, 2020.

Aapo Hyvärinen. Estimation of Non-Normalized Statistical Models by Score Matching. *Journal of Machine Learning Research*, (6), April 2005.

Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv preprint arXiv:1502.03167*, 2015.

Ye Jia, Ron J. Weiss, Fadi Biadsy, Wolfgang Macherey, Melvin Johnson, Zhifeng Chen, and Yonghui Wu. Direct Speech-to-Speech Translation with a Sequence-to-Sequence Model. In *INTERSPEECH*, 2019.

Lauri Juvela, Bajibabu Bollepalli, Vassilis Tsiaras, and Paavo Alku. Glotneta raw waveform model for the glottal excitation in statistical parametric speech synthesis. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(6):1019–1030, 2019.

Nal Kalchbrenner, Erich Elsen, Karen Simonyan, Seb Noury, Norman Casagrande, Edward Lockhart, Florian Stimberg, Aaron van den Oord, Sander Dieleman, and Koray Kavukcuoglu. Efficient Neural Audio Synthesis. In *ICML*, 2018.

Hyeongju Kim, Hyeonseung Lee, Woo Hyun Kang, Sung Jun Cheon, Byoung Jin Choi, and Nam Soo Kim. WaveNODE: A Continuous Normalizing Flow for Speech Synthesis. In *ICML: Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*, 2020.

Sungwon Kim, Sang-Gil Lee, Jongyoon Song, Jaehyeon Kim, and Sungroh Yoon. FloWaveNet: A generative flow for raw audio. In *ICML*, 2019.

Robert Kubichek. Mel-Cepstral Distance Measure for Objective Speech Quality Assessment. In *IEEE PACRIM*, 1993.

Kundan Kumar, Rithesh Kumar, Thibault de Boissiere, Lucas Gestin, Wei Zhen Teoh, Jose Sotelo, Alexandre de Brebisson, Yoshua Bengio, and Aaron Courville. MelGAN: Generative Adversarial Networks for Conditional Waveform Synthesis. In *NeurIPS*, 2019.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. Deterministic Non-Autoregressive Neural Sequence Modeling by Iterative Refinement. In *EMNLP*, 2018.

Lala Li and William Chan. Big Bidirectional Insertion Representations for Documents. In *EMNLP: Workshop of Neural Generation and Translation*, 2019.

Ollie McCarthy and Zohaib Ahmed. HooliGAN: Robust, High Quality Neural Vocoding. *arXiv preprint arXiv:2008.02493*, 2020.

Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron Courville, and Yoshua Bengio. SampleRNN: An Unconditional End-to-End Neural Audio Generation Model. *arXiv preprint arXiv:1612.07837*, 2016.

Aäron 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.

Aäron van den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George van den Driessche, Edward Lockhart, Luis C. Cobo, Florian Stimberg, Norman Casagrande, Dominik Grewe, Seb Noury, Sander Dieleman, Erich Elsen, Nal Kalchbrenner, Heiga Zen, Alex Graves, Helen King, Tom Walters, Dan Belov, and Demis Hassabis. Parallel WaveNet: Fast High-Fidelity Speech Synthesis. In *ICML*, 2018.

Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic Image Synthesis with Spatially-Adaptive Normalization. In *CVPR*, 2019.

Kainan Peng, Wei Ping, Zhao Song, and Kexin Zhao. Non-Autoregressive Neural Text-to-Speech. In *ICML*, 2020.

Wei Ping, Kainan Peng, and Jitong Chen. ClariNet: Parallel Wave Generation in End-to-End Text-to-Speech. In *ICLR*, 2019.

Ryan Prenger, Rafael Valle, and Bryan Catanzaro. WaveGlow: A Flow-based Generative Network for Speech Synthesis. In *ICASSP*, 2019.

Laura Ruis, Mitchell Stern, Julia Proskurnia, and William Chan. Insertion-Deletion Transformer. In *EMNLP: Workshop of Neural Generation and Translation*, 2019.

Sara Sabour, William Chan, and Mohammad Norouzi. Optimal Completion Distillation for Sequence Learning. In *ICLR*, 2019.

Chitwan Saharia, William Chan, Saurabh Saxena, and Mohammad Norouzi. Non-Autoregressive Machine Translation with Latent Alignments. *arXiv preprint arXiv:2004.07437*, 2020.

Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications. In *ICLR*, 2017.

Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact Solutions to the Nonlinear Dynamics of Learning in Deep Linear Neural Networks. In *ICLR*, 2014.

Jonathan Shen, Ruoming Pang, Ron J. Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, RJ Skerrv-Ryan, Rif A. Saurous, Yannis Agiomyrgiannakis, and Yonghui Wu. Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions. In *ICASSP*, 2018.

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In *ICML*, 2015.

Eunwoo Song, Kyungguen Byun, and Hong-Goo Kang. ExcitNet Vocoder: A Neural Excitation Model for Parametric Speech Synthesis Systems. In *EUSIPCO*, 2019a.

Yang Song and Stefano Ermon. Generative Modeling by Estimating Gradients of the Data Distribution. In *NeurIPS*, 2019.

Yang Song and Stefano Ermon. Improved Techniques for Training Score-Based Generative Models. *arXiv preprint arXiv:2006.09011*, 2020.

Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced Score Matching: A Scalable Approach to Density and Score Estimation. *arXiv preprint arXiv:1905.07088*, 2019b.

Jose Sotelo, Soroush Mehri, Kundan Kumar, Joao Felipe Santos, Kyle Kastner, Aaron C. Courville, and Yoshua Bengio. Char2Wav: End-to-End Speech Synthesis. In *ICLR*, 2017.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. Insertion Transformer: Flexible Sequence Generation via Insertion Operations. In *ICML*, 2019.

Jean-Marc Valin and Jan Skoglund. LPCNet: Improving Neural Speech Synthesis through Linear Prediction. In *ICASSP*, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *NIPS*, 2017.

Pascal Vincent. A Connection Between Score Matching and Denoising Autoencoders. *Neural Computation*, 2011.

Xin Wang, Shinji Takaki, and Junichi Yamagishi. Neural Source-Filter Waveform Models for Statistical Parametric Speech Synthesis. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:402–415, 2020.

Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J. Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, Quoc Le, Yannis Agiomyrgiannakis, Rob Clark, and Rif A. Saurous. Tacotron: Towards End-to-End Speech Synthesis. In *INTERSPEECH*, 2017.

Ning-Qian Wu and Zhen-Hua Ling. WaveFFJORD: FFJORD-Based Vocoder for Statistical Parametric Speech Synthesis. In *ICASSP*, 2020.

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.

Jinhyeok Yang, Junmo Lee, Youngik Kim, Hoonyoung Cho, and Injung Kim. VocGAN: A High-Fidelity Real-time Vocoder with a Hierarchically-nested Adversarial Network. *arXiv preprint arXiv:2007.15256*, 2020.