WaveFlow: A Compact Flow-based Model for Raw Audio

Wei Ping [1] **Kainan Peng** [1] **Kexin Zhao** [1] **Zhao Song** [1]

Abstract

In this work, we propose WaveFlow, a small-footprint generative flow for raw audio, which is directly trained with maximum likelihood. It handles the long-range structure of 1-D waveform with a dilated 2-D convolutional architecture, while modeling the local variations using expressive autoregressive functions. WaveFlow provides a unified view of likelihood-based models for 1-D data, including WaveNet and WaveGlow as special cases. It generates high-fidelity speech as WaveNet, while synthesizing several orders of magnitude faster as it only requires a few sequential steps to generate very long waveforms with hundreds of thousands of time-steps. Furthermore, it can significantly reduce the likelihood gap that has existed between autoregressive models and flow-based models for efficient synthesis. Finally, our small-footprint WaveFlow has only 5.91M parameters, which is 15× smaller than WaveGlow. It can generate 22.05 kHz high-fidelity audio 42.6× faster than real-time (at a rate of 939.3 kHz) on a V100 GPU without engineered inference kernels.

1. Introduction

Deep generative models have obtained noticeable successes for modeling raw audio in high-fidelity speech synthesis and music generation (e.g., van den Oord et al., 2016; Dieleman et al., 2018). Autoregressive models are among the best performing generative models for raw waveforms, providing the highest likelihood scores and generating high-fidelity audios (van den Oord et al., 2016; Mehri et al., 2017; Kalchbrenner et al., 2018). One of the most successful examples is WaveNet (van den Oord et al., 2016), an autoregressive model for waveform synthesis. It operates at the high temporal resolution (e.g., 24 kHz) of raw audio and sequentially generates 1-D waveform samples at inference. As a result, WaveNet is prohibitively slow for speech synthesis and one has to develop highly engineered kernels (Arık et al., 2017a; Pharris, 2018) for real-time inference, which is a requirement for most production text-to-speech systems.

Flow-based models (Dinh et al., 2014; Rezende & Mohamed, 2015) are a family of generative models, in which a simple initial density is transformed into a complex one by applying a series of invertible transformations. One group of models are based on *autoregressive transformation*, including autoregressive flow (AF) and inverse autoregressive flow (IAF) as the "dual" of each other (Kingma et al., 2016; Papamakarios et al., 2017; Huang et al., 2018). AF is analogous to autoregressive models, which performs parallel density evaluation and sequential synthesis. In contrast, IAF performs parallel synthesis but sequential density evaluation, making likelihood-based training very slow. Parallel WaveNet (van den Oord et al., 2018) distills an IAF from a pretrained autoregressive WaveNet, which gets the best of both worlds. However, one has to apply the Monte Carlo method to approximate the intractable KL divergence in distillation. Instead, ClariNet (Ping et al., 2019) simplifies the probability density distillation by computing a regularized KL divergence in closed-form. Both of them require a pretrained WaveNet teacher and a set of auxiliary losses for high-fidelity synthesis, which complicates the training pipeline and increases the cost of development.

Another group of flow-based models are based on *bipartite transformation* (Dinh et al., 2017; Kingma & Dhariwal, 2018), which provide likelihood-based training and parallel synthesis. Most recently, WaveGlow (Prenger et al., 2019) and FloWaveNet (Kim et al., 2019) apply Glow (Kingma & Dhariwal, 2018) and RealNVP (Dinh et al., 2017) for waveform synthesis, respectively. However, the bipartite flows require more layers, larger hidden size, and huge number of parameters to reach comparable capacities as autoregressive models. In particular, WaveGlow and FloWaveNet have 87.88M and 182.64M parameters with 96 layers and 256 residual channels, respectively, whereas a typical 30-layer WaveNet has 4.57M parameters with 128 residual channels. Moreover, both of them *squeeze* the time-domain samples on the *channel* dimension before applying the bipartite transformation, which may lose the temporal order information and make them less efficient at modeling waveform sequence.

[1]Baidu Research, 1195 Bordeaux Dr, Sunnyvale, CA.
Project page: https://waveflow-demo.github.io/.
Correspondence to: Wei Ping <weiping.thu@gmail.com>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020.

In this work, we present WaveFlow, a compact flow-based model for raw audio, which features (i) simple training, (ii) high-fidelity & ultra-fast synthesis, and (iii) small footprint. Specifically, we make the following contributions:

1. WaveFlow is directly trained with maximum likelihood without probability density distillation and auxiliary losses as used in Parallel WaveNet (van den Oord et al., 2018) and ClariNet (Ping et al., 2019), which simplifies the training pipeline and reduces the cost of development. WaveFlow squeezes the 1-D waveform samples into a 2-D matrix and processes the local adjacent samples with autoregressive functions without losing temporal order information. We implement WaveFlow with a dilated 2-D convolutional architecture (Yu & Koltun, 2015), which leads to 15× fewer parameters, and faster synthesis speed than WaveGlow (Prenger et al., 2019).
2. WaveFlow provides a unified view of likelihood-based models for raw audio, which includes both WaveNet and WaveGlow as special cases and allows us to explicitly trade inference parallelism for model capacity. We systematically study these models in terms of test likelihood and audio fidelity. We demonstrate that a moderate-sized WaveFlow can obtain comparable likelihood and synthesize high-fidelity speech as WaveNet, while synthesizing thousands of times faster. In previous work, there is a large likelihood gap between autoregressive models and flow-based models which provide efficient sampling (Ho et al., 2019; Tran et al., 2019).
3. For practitioners, our small WaveFlow has only 5.91M parameters by utilizing the compact autoregressive functions for modeling local signal variations. It synthesizes 22.05 kHz high-fidelity speech (MOS: 4.32) more than 40× faster than real-time on a Nvidia V100 GPU. In contrast, WaveGlow (Prenger et al., 2019) requires 87.88M parameters for generating high-fidelity speech. The small memory footprint is preferred in production TTS systems, especially for on-device deployment.

We organize the rest of the paper as follows. Section 2 reviews the flow-based models. We present WaveFlow in Section 3 and discuss related work in Section 4. We report experimental results in Section 5 and discuss the pros and cons of different methods in Section 6.

2. Flow-based generative models

Flow-based models (Dinh et al., 2014; 2017; Rezende & Mohamed, 2015) transform a simple density $p(z)$ (e.g., isotropic Gaussian) into a complex data distribution $p(x)$ by applying a bijection $x = f(z)$, where x and z are both n-dimensional. The probability density of x can be obtained through the change of variables formula:

$$p(x) = p(z) \left| \det \frac{\partial f^{-1}(x)}{\partial x} \right|; \quad (1)$$

where $z = f^{-1}(x)$ is the inverse of the bijection, and $\det \frac{\partial f^{-1}(x)}{\partial x}$ is the determinant of its Jacobian. In general, it takes $O(n^3)$ to compute the determinant, which is not scalable in high-dimension. There are two notable groups of flow-based models with triangular Jacobians and tractable determinants, which are based on autoregressive and bipartite transformations, respectively. Before delving into details, we summarize the model capacities and parallelisms of after-mentioned flow-based models in Table 1.

2.1. Autoregressive transformation

The autoregressive flow (AF) and inverse autoregressive flow (IAF) (Kingma et al., 2016; Papamakarios et al., 2017) use autoregressive transformations. Specifically, AF defines $z = f^{-1}(x; \theta)$:

$$z_t = x_t \quad _t(x_{<t}; \theta) + \quad _t(x_{<t}; \theta); \quad (2)$$

where the shifting variables $_t(x_{<t}; \theta)$ and scaling variables $_t(x_{<t}; \theta)$ are modeled by an autoregressive architecture parameterized by θ (e.g., WaveNet). Note that, the t-th variable z_t only depends on $x_{\quad t}$, thus the Jacobian is a triangular matrix as illustrated in Figure 1(a). Its determinant is the product of the diagonal entries: $\det \frac{\partial f^{-1}(x)}{\partial x} = \prod_t \quad _t(x_{<t}; \theta)$. The density $p(x)$ can be evaluated in parallel by formula Eq.(1), because the minimum number of sequential operations is $O(1)$ for computing $z = f^{-1}(x)$ (see Table 1). However, AF has to do sequential synthesis, because $x = f(z)$ is autoregressive: $x_t = \frac{z_t \quad _t(x_{<t}; \theta)}{_t(x_{<t}; \theta)}$. Note that, Gaussian autoregressive model can be equivalently interpreted as an autoregressive flow (Kingma et al., 2016).

In contrast, IAF uses an autoregressive transformation for inverse mapping $z = f^{-1}(x)$:

$$z_t = \frac{x_t \quad _t(z_{<t}; \theta)}{_t(z_{<t}; \theta)}; \quad (3)$$

making density evaluation very slow for likelihood-based training, but one can sample $x = f(z)$ in parallel through $x_t = z_t \quad _t(z_{<t}; \theta) + \quad _t(z_{<t}; \theta)$. Parallel WaveNet (van den Oord et al., 2018) and ClariNet (Ping et al., 2019) are based on IAF for parallel synthesis, and they rely on the probability density distillation from a pretrained autoregressive WaveNet at training.

2.2. Bipartite transformation

RealNVP (Dinh et al., 2017) and Glow (Kingma & Dhariwal, 2018) use bipartite transformation by partitioning the data x into two groups x_a and x_b, where the indices sets $a \cup b = \{1, \ldots, n\}$ and $a \cap b = \emptyset$. Then, the inverse mapping $z = f^{-1}(x; \theta)$ is defined as:

$$z_a = x_a; \quad z_b = x_b \quad _b(x_a; \theta) + \quad _b(x_a; \theta): \quad (4)$$

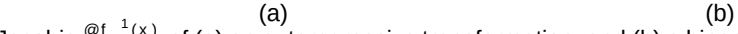

Figure 1. The Jacobian $\frac{\partial f^{-1}(x)}{\partial x}$ of (a) an autoregressive transformation, and (b) a bipartite transformation. The blank cells are zeros and represent the independent relations between z_i and x_j. The light-blue cells with scaling variables represent the linear dependencies. The dark-blue cells represent complex non-linear dependencies.

Table 1. The minimum number of sequential operations (indicates parallelism) required by flow-based models for density evaluation $z = f^{-1}(x)$ and sampling $x = f(z)$. Therein, n is the length of x, h is the squeezed height in WaveFlow. In WaveFlow, larger h leads to higher model capacity, but more sequential steps for sampling.

Flow-based model	Sequential operations for $z = f^{-1}(x)$	Sequential operations for $x = f(z)$	Model capacity (same size)
AF	O(1)	O(n)	high
IAF	O(n)	O(1)	high
Bipartite flow	O(1)	O(1)	low
WaveFlow	O(1)	O(h)	low $\sim$ high

where the shifting variables $\mu_b(x_a; \theta)$ and scaling variables $\sigma_b(x_a; \theta)$ are modeled by a feed-forward neural network. Its Jacobian $\frac{\partial f^{-1}(x)}{\partial x}$ is a special triangular matrix as illustrated in Figure 1(b). By definition, $x = f(z; \theta)$ is,

$$x_a = z_a; \quad x_b = \frac{z_b - \mu_b(x_a; \theta)}{\sigma_b(x_a; \theta)}. \tag{5}$$

Note that, both evaluating $z = f^{-1}(x; \theta)$ and sampling $x = f(z; \theta)$ can be done in parallel.

WaveGlow (Prenger et al., 2019) and FloWaveNet (Kim et al., 2019) squeeze the time-domain samples on the channel dimension, then apply the bipartite transformation on the partitioned channels. Note that, this squeezing operation is inefficient, as one may lose the temporal order information. As a result of doing this, the synthesized audio can have constant frequency noises (see Appendix A for an example).

2.3. Connections

The autoregressive transformation is more expressive than bipartite transformation. As illustrated in Figure 1(a) and (b), the autoregressive transformation introduces $\frac{n(n-1)}{2}$ complex non-linear dependencies (dark-blue cells) and n linear dependencies between data x and latents z. In contrast, bipartite transformation has only $\frac{n^2}{4}$ non-linear dependencies and $\frac{n}{2}$ linear dependencies. Indeed, one can easily reduce an autoregressive transformation $z = f^{-1}(x; \theta)$ to a bipartite transformation $z = f^{-1}(x; \theta)$ by: (i) picking an autoregressive order o such that all of the indices in set a rank early than the indices in b, and (ii) setting the shifting and scaling variables as,

$$\begin{pmatrix} \mu_t(x_{<t}; \theta) \\ \sigma_t(x_{<t}; \theta) \end{pmatrix} = \begin{cases} [0, 1]^\top, & \text{for } t \in a \\ [\mu_t(x_a; \theta), \sigma_t(x_a; \theta)]^\top, & \text{for } t \in b \end{cases}.$$

Given the less expressive building block, the bipartite flows require more layers and larger hidden size to reach the capacity of autoregressive models (e.g., measured by likelihood).

3. WaveFlow

In this section, we present WaveFlow and its implementation with dilated 2-D convolutions. We also discuss the permutation strategies for stacking multiple flows.

3.1. Definition

We denote the 1-D waveform as $x = \{x_1, \ldots, x_n\}$. We squeeze x into an h-row 2-D matrix $X \in \mathbb{R}^{h \times w}$ by column-major order, where the adjacent samples are in the same column. We assume $Z \in \mathbb{R}^{h \times w}$ are sampled from an isotropic Gaussian distribution, and define $Z = f^{-1}(X; \theta)$ as,

$$Z_{i,j} = \sigma_{i,j}(X_{<i,\cdot}; \theta) \cdot X_{i,j} + \mu_{i,j}(X_{<i,\cdot}; \theta), \tag{6}$$

where $X_{<i,\cdot}$ represents all elements above i-th row (see Figure 2 for an illustration). Note that, (i) In WaveFlow, the receptive field over the squeezed input X for computing $Z_{i,j}$ is strictly larger than that of WaveGlow when $h > 2$. (ii) WaveNet is equivalent to an autoregressive flow (AF)

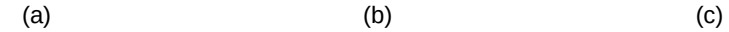

Figure 2. The receptive fields over the squeezed inputs X for computing $Z_{i,j}$ in (a) WaveFlow, (b) WaveGlow, and (c) autoregressive flow with column-major order (e.g., WaveNet).

with the column-major order on X. (iii) Both WaveFlow and WaveGlow look at future waveform samples in original x for computing $Z_{i,j}$, whereas WaveNet can not.

The shifting variables $\mu_{i,j}(X_{<i,:}; \theta)$ and scaling variables $\sigma_{i,j}(X_{<i,:}; \theta)$ in Eq. (6) are modeled by a 2-D convolutional neural network detailed in Section 3.2. By definition, the variable $Z_{i,j}$ only depends on the current $X_{i,j}$ and previous $X_{<i,:}$ in raw-major order, thus the Jacobian is a triangular matrix and its determinant is:

$$\det \frac{\partial f^{-1}(X)}{\partial X} = \prod_{i=1}^{h} \prod_{j=1}^{w} \sigma_{i,j}(X_{<i,:}; \theta) : \quad (7)$$

As a result, the log-likelihood can be calculated in parallel by change of variable formula in Eq. (1),

$$\log p(X) = \sum_{i,j} \log \sigma_{i,j}(X_{<i,:}; \theta) - \frac{Z_{i,j}^2}{2} - \frac{1}{2}\log(2\pi) ;$$

and one can do maximum likelihood training efficiently. At synthesis, we sample Z from the isotropic Gaussian and apply the forward mapping $X = f(Z; \theta)$:

$$X_{i,j} = \frac{Z_{i,j} - \mu_{i,j}(X_{<i,:}; \theta)}{\sigma_{i,j}(X_{<i,:}; \theta)}; \quad (8)$$

which is autoregressive over the height dimension and requires h sequential steps to generate the whole X. In practice, a small h (e.g., 8 or 16) works well, thus we can generate very long waveforms within a few sequential steps.

3.2. Implementation with dilated 2-D convolutions

We implement WaveFlow with a dilated 2-D convolutional architecture. Specifically, we use a stack of 2-D convolution layers (e.g., 8 layers in all our experiments) to model the shifting $\mu_{i,j}(X_{<i,:}; \theta)$ and scaling variables $\sigma_{i,j}(X_{<i,:}; \theta)$ in Eq. (6). We use the similar architecture as WaveNet (van den Oord et al., 2016) by replacing the dilated 1-D convolution with 2-D convolution (Yu & Koltun, 2015), while still keeping the gated-tanh nonlinearities, residual connections and skip connections.

We set the filter sizes as 3 for both height and width dimensions. We use non-causal convolutions on width dimension and set the dilation cycle as $[1, 2, 4, \ldots, 2^7]$. The convolutions on height dimension are causal with the autoregressive constraint, and their dilation cycle needs to be designed carefully. In practice, we find the following rules of thumb are important to obtain good results:

- As motivated by the dilation cycle of WaveNet (van den Oord et al., 2016), the dilations of 8 layers should be set as $d = [1, 2, \ldots, 2^s, 1, 2, \ldots, 2^s, \ldots]$, where $s = 7$. [1]
- The receptive field over the height dimension should be larger than or equal to height h. Otherwise, it introduces unnecessary conditional independence and leads to lower likelihood (see Table 2 for an example). Note that, the receptive field of a stack of dilated convolutional layers is $r = (k-1)\sum_i d_i + 1$, where k is the filter size and d_i is the dilation at i-th layer. Thus, the sum of dilations should satisfy: $\sum_i d_i \geq \frac{h-1}{k-1}$. However, when h is larger than or equal to $2^8 = 512$, we simply set the dilation cycle as $[1, 2, 4, \ldots, 2^7]$. When r has already been larger than h, the convolutions with smaller dilations provide larger likelihood.

We summarize the heights and preferred dilations in our experiments in Table 3. We also implement convolution queue (Paine et al., 2016) to cache the intermediate hidden states, which will speed up the autoregressive inference over the height dimension. Note that, WaveFlow is fully autoregressive and equivalent to a Gaussian WaveNet (Ping et al., 2019), when we squeeze x by its length (i.e., $h = n$) and set its filter size as 1 over the width dimension. If we squeeze x by $h = 2$ and set the filter size as 1 on height dimension, WaveFlow becomes a bipartite flow and is equivalent to a WaveGlow with squeezed channels 2.

3.3. Local conditioning for speech synthesis

In neural speech synthesis, a neural vocoder (e.g., WaveNet) synthesizes the time-domain waveforms, which can be conditioned on linguistic features (van den Oord et al., 2016; Arik et al., 2017a), the mel spectrograms from a text-to-spectrogram model (Ping et al., 2018; Shen et al., 2018), or the learned hidden representation within a text-to-wave architecture (Ping et al., 2019). In this work, we test WaveFlow by conditioning it on ground-truth mel spectrograms as in previous work (Prenger et al., 2019; Kim et al., 2019). The mel spectrogram is upsampled to the same length as

[1] We did try different setups, but they all lead to worse likelihood scores.

Table 2. The test log-likelihoods (LLs) of WaveFlow with different dilation cycles on the height dimension when $h = 32$. The models are stacked with 8 flows and each flow has 8 layers.

Model	Res. channels c	Dilations d	Receptive field r	Test LLs
WaveFlow ($h = 32$)	128	1, 1, 1, 1, 1, 1, 1, 1	17	4.960
WaveFlow ($h = 32$)	128	1, 2, 4, 1, 2, 4, 1, 2	35	5.055

Table 3. The height h, filter size k over the height dimension, and the corresponding dilations used in our experiments. Note that, the receptive fields r are only slightly larger than heights h.

h	k	Dilations d	Receptive field r
8	3	1, 1, 1, 1, 1, 1, 1, 1	17
16	3	1, 1, 1, 1, 1, 1, 1, 1	17
32	3	1, 2, 4, 1, 2, 4, 1, 2	35
64	3	1, 2, 4, 8, 16, 1, 2, 4	77

waveform samples with transposed 2-D convolutions (Ping et al., 2019). To be aligned with the waveform, they are squeezed to the shape $h \times w$, where c is the input channels (e.g, mel bands). After a 1×1 convolution mapping the input channels to residual channels, they are added as the bias term at each layer (Ping et al., 2019).

3.4. Stacking multiple flows with permutations on height dimension

Flow-based models require a series of transformations until the distribution $p(X)$ reaches a desired level of capacity. We denote $X = Z^{(n)}$ and repeatedly apply the transformation $Z^{(i-1)} = f^{-1}(Z^{(i)}; \theta^{(i)})$ defined in Eq.(6) from $Z^{(n)}$ to $Z^{(0)}$, where $Z^{(0)}$ are from the isotropic Gaussian. Thus, the $p(X)$ can be evaluated by applying the chain rule:

$$p(X) = p(Z^{(0)}) \prod_{i=1}^{n} \det \left| \frac{\partial f^{-1}(Z^{(i)}; \theta^{(i)})}{\partial Z^{(i)}} \right| .$$

We find that permuting each $Z^{(i)}$ over its height dimension after each transformation can significantly improve the likelihood scores. In particular, we test two permutation strategies for WaveFlow models stacked with 8 flows (i.e., $X = Z^{(8)}$) in Table 4: a) we reverse each $Z^{(i)}$ over the height dimension after each transformation, and b) we reverse $Z^{(7)}$, $Z^{(6)}$, $Z^{(5)}$, $Z^{(4)}$ over the height dimension as before, but bipartition[2] $Z^{(3)}$, $Z^{(2)}$, $Z^{(1)}$, $Z^{(0)}$ in the middle of the height dimension then reverse each part respectively. In speech synthesis, one needs to permute the conditioner accordingly over the height dimension, which is aligned with $Z^{(i)}$. In Table 4, both strategies a) and b) significantly outperform the model without permutations mainly because of bidirectional modeling. Strategy b) outperforms a), and we attribute this to its diverse autoregressive orders.

[2] After bipartition & reverse, the height dimension $[0, \cdots, \frac{h}{2}-1, \frac{h}{2}, \cdots, h-1]$ becomes $[\frac{h}{2}-1, \cdots, 0, h-1, \cdots, \frac{h}{2}]$.

4. Related work

Neural speech synthesis has obtained the state-of-the-art results and received a lot of attention. Several neural text-to-speech (TTS) systems have been introduced, including WaveNet (van den Oord et al., 2016), Deep Voice 1 & 2 & 3 (Arik et al., 2017a;b; Ping et al., 2018), Tacotron 1 & 2 (Wang et al., 2017; Shen et al., 2018), Char2Wav (Sotelo et al., 2017), VoiceLoop (Taigman et al., 2018), WaveRNN (Kalchbrenner et al., 2018), ClariNet (Ping et al., 2019), Transformer TTS (Li et al., 2019), ParaNet (Peng et al., 2019), and FastSpeech (Ren et al., 2019).

Neural vocoders (waveform synthesizer), such as WaveNet, play the most important role in recent advances of speech synthesis. In previous work, the state-of-the-art neural vocoders are autoregressive models (van den Oord et al., 2016; Mehri et al., 2017; Kalchbrenner et al., 2018). Several endeavors have been advocated for speeding up their sequential generation process (Arik et al., 2017a; Kalchbrenner et al., 2018). In particular, Subscale WaveRNN (Kalchbrenner et al., 2018) folds a long waveform sequence $x_{1:n}$ into a batch of shorter sequences and can produce $\frac{n}{16}$ samples per step, thus it requires at least $\frac{n}{16}$ steps to generate the whole audio. This is different from WaveFlow, which can generate $x_{1:n}$ within e.g. 16 steps.

Flow-based models can either represent the approximate posteriors for variational inference (Rezende & Mohamed, 2015; Kingma et al., 2016; Berg et al., 2018), or can be trained directly on data using the change of variables formula (Dinh et al., 2014; 2017; Kingma & Dhariwal, 2018). We consider the later case in this work. In previous work, Glow (Kingma & Dhariwal, 2018) extends RealNVP (Dinh et al., 2017) with invertible 1×1 convolution on channel dimension, which first generates high-fidelity images. Hoogeboom et al. (2019) generalizes the invertible convolution to operate on both channels and spatial axes. Most recently, flow-based models have been successfully applied for parallel waveform synthesis with comparable fidelity as autoregressive models (van den Oord et al., 2018; Ping et al., 2019; Yamamoto et al., 2019b; Prenger et al., 2019; Kim et al., 2019; Serrà et al., 2019). Among these models, WaveGlow (Prenger et al., 2019) and FloWaveNet (Kim et al., 2019) have a simple training pipeline as they solely use the maximum likelihood objective. However, both of them are less expressive than autoregressive models as indicated by their large memory footprint and lower likelihood scores.

Table 4. The test LLs of WaveFlow with different permutation strategies. All models consist of 8 flows and each flow has 8 convolutional layers with filter sizes 3.

Model	Res. channels	Permutation strategy	Test LLs
WaveFlow (h = 16)	64	none	4.551
WaveFlow (h = 16)	64	a) 8 reverse	4.954
WaveFlow (h = 16)	64	b) 4 reverse, 4 bipartition & reverse	4.971

5. Experiment

In this section, we compare likelihood-based generative models for raw audio in term of test likelihood, synthesis speed and speech fidelity.[3] The results in this section are obtained from an internal PyTorch implementation. We provide a PaddlePaddle reimplementation in Parakeet toolkit.[4]

Data: We use the LJ speech dataset (Ito, 2017) containing about 24 hours of audio with a sampling rate of 22.05 kHz recorded on a MacBook Pro in a home enviroment. It consists of 13,100 audio clips from a single female speaker.

Models: We evaluate several likelihood-based models, including WaveFlow, Gaussian WaveNet (Ping et al., 2019), WaveGlow (Prenger et al., 2019), and autoregressive flow (AF). As illustrated in Section 3.2, we implement AF from WaveFlow by squeezing the waveforms by its length and setting the filter size as 1 over width dimension. Both WaveNet and AF have 30 layers with dilation cycle $[1, 2, \ldots, 512]$ and filter size 3. For WaveFlow and WaveGlow, we investigate different setups, including the number of flows, size of residual channels, and squeezed height.

Conditioner: We use the 80-band mel spectrogram of the original audio as the conditioner for WaveNet, WaveGlow, and WaveFlow. We set FFT size to 1024, hop size to 256, and window size to 1024. For WaveNet and WaveFlow, we upsample the mel conditioner 256 times by applying two layers of transposed 2-D convolution (in time and frequency) interleaved with leaky ReLU ($\alpha = 0.4$) (Ping et al., 2019). The upsampling strides in time are 16 and the 2-D convolution filter sizes are $[32, 3]$ for both layers. In WaveFlow, the upsampled mel spectrogram is squeezed along the temporal dimension as waveform and its shape becomes [mel-band, height, width]. After that, we apply 1x1 conv to map its channels from mel-band to the residual channel in each 2-D convolution layer. Finally, it is added as bias term within the dilated convolution operation before the gated-tanh nonlinearities, which is the same as WaveNet For WaveGlow, we directly use Nvidia's open source implementation.

Training: We train all models on 8 Nvidia 1080Ti GPUs using randomly chosen short clips of 16,000 samples from each utterance. For WaveFlow and WaveNet, we use the Adam optimizer (Kingma & Ba, 2015) with a batch size of 8 and a constant learning rate of 2×10^{-4}. For WaveGlow, we use the Adam optimizer with a batch size of 16 and a learning rate of 1×10^{-4}. We apply weight normalization (Salimans & Kingma, 2016) whenever possible.

5.1. Likelihood

We evaluate the test log-likelihoods (LLs) of WaveFlow, WaveNet, WaveGlow and autoregressive flow (AF) conditioned on mel spectrograms at 1M training steps. We choose 1M steps as the cut-off, because the LLs decrease slowly after that, and it already took one month to train the largest WaveGlow (residual channels = 512) for 1M steps. We summarize the results in Table 5 with models from row a) to row t). We draw the following observations:

- Stacking a large number of flows improves LLs for all flow-based models. For example, WaveFlow (h = 16) with 8 flows provides larger LL than WaveFlow (h = 16) with 6 flows. The autoregressive flow (b) obtains the highest likelihood and outperforms WaveNet (a) with the same amount of parameters. Indeed, AF provides bidirectional modeling by stacking 3 flows with reverse operations.
- WaveFlow has much larger likelihood than WaveGlow with comparable number of parameters. In particular, a small-footprint WaveFlow (k) has only 5.91M parameters but can provide comparable likelihood (5.023 vs. 5.026) as the largest WaveGlow (g) with 268.29M parameters.
- As we increase h, the likelihood of WaveFlow steadily increases (can be seen from (h)-(k)), and its inference will be slower on GPU with more sequential steps. In the limit, it is equivalent to an AF. It illustrates the trade-off between model capacity and inference parallelism.
- WaveFlow (r) with 128 residual channels can obtain comparable likelihood (5.055 vs 5.059) as WaveNet (a) with 128 residual channels. A larger WaveFlow (s) with 256 residual channels can obtain even larger likelihood than WaveNet (5.101 vs 5.059).

Note that, there is a significant likelihood gap that has so far existed between autoregressive models and flow-based models providing efficient sampling (e.g., Ho et al., 2019; Tran et al., 2019). WaveFlow can close the likelihood gap with a modest squeezing height, which suggests the strength of autoregressive model is mainly at modeling the local structure of the signal.

[3] Speech samples are in https://waveflow-demo.github.io/

[4] https://github.com/PaddlePaddle/Parakeet/tree/develop/examples/waveflow

Table 5. The test log-likelihoods (LLs) of all models conditioned on mel spectrograms. For $a \times b = c$ in the "flows × layers" column, a is number of flows, b is number of layers in each flow, and c is the total number of layers. In WaveFlow, h is the squeezed height. Models with bolded test LLs are mentioned in the text.

	Model	flows × layers	Res. channels	# Param	Test LLs
(a)	Gaussian WaveNet	1 × 30 = 30	128	4.57 M	5.059
(b)	Autoregressive flow	3 × 10 = 30	128	4.54 M	5.161
(c)	WaveGlow	12 × 8 = 96	64	17.59 M	4.804
(d)	WaveGlow	12 × 8 = 96	128	34.83 M	4.927
(e)	WaveGlow	6 × 8 = 48	256	47.22 M	4.922
(f)	WaveGlow	12 × 8 = 96	256	87.88 M	5.018
(g)	WaveGlow	12 × 8 = 96	512	268.29 M	5.026
(h)	WaveFlow ($h = 8$)	8 × 8 = 64	64	5.91 M	4.935
(i)	WaveFlow ($h = 16$)	8 × 8 = 64	64	5.91 M	4.954
(j)	WaveFlow ($h = 32$)	8 × 8 = 64	64	5.91 M	5.002
(k)	WaveFlow ($h = 64$)	8 × 8 = 64	64	5.91 M	5.023
(l)	WaveFlow ($h = 8$)	6 × 8 = 48	96	9.58 M	4.946
(m)	WaveFlow ($h = 8$)	8 × 8 = 64	96	12.78 M	4.977
(n)	WaveFlow ($h = 16$)	8 × 8 = 64	96	12.78 M	5.007
(o)	WaveFlow ($h = 16$)	6 × 8 = 48	128	16.69 M	4.990
(p)	WaveFlow ($h = 8$)	8 × 8 = 64	128	22.25 M	5.009
(q)	WaveFlow ($h = 16$)	8 × 8 = 64	128	22.25 M	5.028
(r)	WaveFlow ($h = 32$)	8 × 8 = 64	128	22.25 M	5.055
(s)	WaveFlow ($h = 16$)	6 × 8 = 48	256	64.64 M	5.064
(t)	WaveFlow ($h = 16$)	8 × 8 = 64	256	86.18 M	5.101

5.2. Speech fidelity and synthesis speed

We use the permutation strategy described in Table 4 for WaveFlow. We train WaveNet for 1M steps. We train large WaveGlow and WaveFlow (res. channels 256 and 512) for 1M steps due to the practical time constraint. We train moderate size models (res. channels 128) for 2M steps. We train small size models (res. channels 64 and 96) for 3M steps with slightly improved performance after 2M steps. We use the same setting of ClariNet as in Ping et al. (2019). At synthesis, we sample z from an isotropic Gaussian with standard deviation 1.0 and 0.6 (default) for WaveFlow and WaveGlow, respectively. We use the crowdMOS toolkit (Ribeiro et al., 2011) for speech quality evaluation, where test utterances from these models were presented to workers on Mechanical Turk. In addition, we test the synthesis speed on a Nvidia V100 GPU without using any engineered inference kernels. For WaveFlow and WaveGlow, we run synthesis under NVIDIA Apex with 16-bit floating point (FP16) arithmetic, which does not introduce any degradation of audio fidelity and brings about 2× speedup. We implement convolution queue (Paine et al., 2016) in Python to cache the intermediate hidden states in WaveFlow for autoregressive inference over the height dimension, which brings another 3× to 5× speedup depending on height h.

We report the 5-scale Mean Opinion Score (MOS), synthesis speed and model footprint in Table 6. We draw the following observations:

The small WaveFlow (res. channels 64) has 5.91M parameters, and can synthesize 22.05 kHz high-fidelity speech (MOS: 4.32) 42.60× faster than real-time. In contrast, the speech quality of small WaveGlow (res. channels 64) is significantly worse (MOS: 2.17). Indeed, WaveGlow (res. channels 256) requires 87.88M parameters for generating high-fidelity speech.

The large WaveFlow (res. channels 256) outperforms the same size WaveGlow in terms of speech fidelity (MOS: 4.43 vs. 4.34). It also matches the state-of-the-art WaveNet while generating speech 6.14× faster than real-time, because it only requires 128 sequential steps (number of flows × height) to synthesize very long waveforms with hundreds of thousands time-steps.

ClariNet has the smallest footprint and provides reasonably good speech fidelity (MOS: 4.22) because of its "mode seeking" behavior. In contrast, likelihood-based models are forced to model all possible variations existing in the data, which can lead to higher fidelity samples as long as they have enough model capacity.

We also note a positive correlation between the test likelihoods and MOS scores for likelihood-based models (see Figure 3). The larger LLs roughly correspond to higher MOS scores even when we compare all models. This correlation becomes even more evident when we consider each model separately. It suggests that one may use the likelihood score as an objective measure for model selection.

Table 6. The model size, synthesis speed over real-time, and the 5-scale Mean Opinion Scores (MOS) with 95% confidence intervals.

Model	flows	layers	res. channels	# param	syn. speed	MOS
Gaussian WaveNet	1	30 = 30	128	4.57 M	0.002	4.43 ± 0.14
ClariNet	6	10 = 60	64	2.17 M	21.64	4.22 ± 0.15
WaveGlow	12	8 = 96	64	17.59 M	93.53	2.17 ± 0.13
WaveGlow	12	8 = 96	128	34.83 M	69.88	2.97 ± 0.15
WaveGlow	12	8 = 96	256	87.88M	34.69	4.34 ± 0.11
WaveGlow	12	8 = 96	512	268.29 M	8.08	4.32 ± 0.12
WaveFlow ($h = 8$)	8	8 = 64	64	5.91 M	47.61	4.26 ± 0.12
WaveFlow ($h = 16$)	8	8 = 64	64	5.91M	42.60	4.32 ± 0.08
WaveFlow ($h = 16$)	8	8 = 64	96	12.78 M	26.23	4.34 ± 0.13
WaveFlow ($h = 16$)	8	8 = 64	128	22.25 M	21.32	4.38 ± 0.09
WaveFlow ($h = 16$)	8	8 = 64	256	86.18 M	8.42	4.43 ± 0.10
Ground-truth		—	—	—	—	4.56 ± 0.09

Figure 3. The test log-likelihoods (LLs) vs. MOS scores for all likelihood-based models in Table 6.

Table 7. MOS ratings with 95% confidence intervals in text-to-speech experiments.

Method	MOS
Deep Voice 3 + WaveNet	4.21 ± 0.08
Deep Voice 3 + WaveGlow	3.98 ± 0.11
Deep Voice 3 + WaveFlow	4.17 ± 0.09

5.3. Text-to-Speech

We also test WaveFlow for text-to-speech on a proprietary dataset for convenient reasons. It contains 20 hours of audio from a female speaker with a sampling rate of 24 kHz. We use Deep Voice 3 (DV3) (Ping et al., 2018) to predict mel spectrograms from text. We train a 20-layer WaveNet (res. channel = 256, # param = 9.08 M)[5], WaveGlow (# param = 87.88 M), and WaveFlow ($h = 16$, # param = 5.91 M) which are conditioned on teacher-forced mel spectrograms from DV3. For WaveGlow, we apply the denoising function with strength 0.1 in the repository to alleviate the constant frequency noise in synthesized audio. For WaveFlow, we sample Z from isotropic Gaussian with standard deviation 0.95 to counteract the mismatch of mel conditioners between teacher-forced training and autoregressive inference from DV3. We report the MOS results in Table 7. As a result, WaveFlow is a very compelling neural vocoder, which features *i*) simple likelihood-based training, *ii*) high-fidelity & ultra-fast synthesis, and *iii*) small memory footprint.

[5] The WaveNet hyperparameters were tuned for internal data.

6. Discussion

Parallel WaveNet and ClariNet minimize the reverse KL divergence (KLD) between the student and teacher models in probability density distillation, which has the "mode seeking" behavior and leads to whisper voices in practice. As a result, several auxiliary losses are introduced to alleviate the problem, including STFT loss, perceptual loss, contrastive loss and adversarial loss (van den Oord et al., 2018; Ping et al., 2019; Wang et al., 2019; Yamamoto et al., 2019b). In practice, it complicates the system tuning and increases the cost of development. Since it does not need to model the numerous modes in real data distribution, a small-footprint model can generate good quality speech, when the auxiliary losses are carefully tuned. It is worth mentioning that GAN-based models also exhibit similar "mode seeking" behavior for speech synthesis (Kumar et al., 2019; Bińkowski et al., 2019; Yamamoto et al., 2019a). In contrast, likelihood-based models (WaveFlow, WaveGlowl WaveNet) minimize the forward KLD between the model and data distribution. Because the model is forced to learn all possible modes within the real data, the synthesized audio can be very realistic with enough model capacity. However, when the model does not have enough capacity, its performance degrades quickly due to the "mean seeking" behavior of forward KLD (e.g., WaveGlow with 128 res. channels).

Although audio signals are mostly dominated by low-frequency components (e.g., in terms of amplitude), human ears are very sensitive to high-frequency content. As a result, it is crucial to accurately model the local variations of waveform for high-fidelity synthesis, which is indeed the strength of autoregressive models. However, autoregressive models are less efficient at modeling long-range correlations, which can be seen from the difficulties to generate globally consistent images (Van den Oord et al., 2016; Menick & Kalchbrenner, 2018). Worse still, they are also noticeably slow at synthesis. Non-autoregressive convo-

lutional architectures can do speedy synthesis and easily capture the long-range structure in the data (Radford et al., 2015; Brock et al., 2018), but it could produce spurious high-frequency components which will hurt the audio fidelity (e.g., Donahue et al., 2018). In this work, WaveFlow compactly models the local variations using short-range autoregressive functions, and handles the long-range correlations with a non-autoregressive convolutional architecture, which obtains the best of both worlds.

References

Arik, S. O., Chrzanowski, M., Coates, A., Diamos, G., Gibiansky, A., Kang, Y., Li, X., Miller, J., Raiman, J., Sengupta, S., and Shoeybi, M. Deep Voice: Real-time neural text-to-speech. In *ICML*, 2017a.

Arik, S. O., Diamos, G., Gibiansky, A., Miller, J., Peng, K., Ping, W., Raiman, J., and Zhou, Y. Deep Voice 2: Multi-speaker neural text-to-speech. In *NIPS*, 2017b.

Berg, R. v. d., Hasenclever, L., Tomczak, J. M., and Welling, M. Sylvester normalizing flows for variational inference. *arXiv preprint arXiv:1803.05649*, 2018.

Bińkowski, M., Donahue, J., Dieleman, S., Clark, A., Elsen, E., Casagrande, N., Cobo, L. C., and Simonyan, K. High fidelity speech synthesis with adversarial networks. *arXiv preprint arXiv:1909.11646*, 2019.

Brock, A., Donahue, J., and Simonyan, K. Large scale GAN training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

Dieleman, S., van den Oord, A., and Simonyan, K. The challenge of realistic music generation: modelling raw audio at scale. In *NeurIPS*, 2018.

Dinh, L., Krueger, D., and Bengio, Y. NICE: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using Real NVP. In *ICLR*, 2017.

Donahue, C., McAuley, J., and Puckette, M. Adversarial audio synthesis. *arXiv preprint arXiv:1802.04208*, 2018.

Ho, J., Chen, X., Srinivas, A., Duan, Y., and Abbeel, P. Flow++: Improving flow-based generative models with variational dequantization and architecture design. *arXiv preprint arXiv:1902.00275*, 2019.

Hoogeboom, E., Berg, R. v. d., and Welling, M. Emerging convolutions for generative normalizing flows. *arXiv preprint arXiv:1901.11137*, 2019.

Huang, C.-W., Krueger, D., Lacoste, A., and Courville, A. Neural autoregressive flows. *arXiv preprint arXiv:1804.00779*, 2018.

Ito, K. The LJ speech dataset. 2017.

Kalchbrenner, N., Elsen, E., Simonyan, K., Noury, S., Casagrande, N., Lockhart, E., Stimberg, F., Oord, A. v. d., Dieleman, S., and Kavukcuoglu, K. Efficient neural audio synthesis. In *ICML*, 2018.

Kim, S., Lee, S.-g., Song, J., and Yoon, S. FloWaveNet: A generative flow for raw audio. In *ICML*, 2019.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Kingma, D. P. and Dhariwal, P. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pp. 10215–10224, 2018.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. Improving variational inference with inverse autoregressive flow. In *NIPS*, 2016.

Kumar, K., Kumar, R., de Boissiere, T., Gestin, L., Teoh, W. Z., Sotelo, J., de Brébisson, A., Bengio, Y., and Courville, A. C. Melgan: Generative adversarial networks for conditional waveform synthesis. In *Advances in Neural Information Processing Systems*, pp. 14881–14892, 2019.

Li, N., Liu, S., Liu, Y., Zhao, S., Liu, M., and Zhou, M. Neural speech synthesis with transformer network. *AAAI*, 2019.

Mehri, S., Kumar, K., Gulrajani, I., Kumar, R., Jain, S., Sotelo, J., Courville, A., and Bengio, Y. SampleRNN: An unconditional end-to-end neural audio generation model. In *ICLR*, 2017.

Menick, J. and Kalchbrenner, N. Generating high fidelity images with subscale pixel networks and multidimensional upscaling. *arXiv preprint arXiv:1812.01608*, 2018.

Paine, T. L., Khorrami, P., Chang, S., Zhang, Y., Ramachandran, P., Hasegawa-Johnson, M. A., and Huang, T. S. Fast wavenet generation algorithm. *arXiv preprint arXiv:1611.09482*, 2016.

Papamakarios, G., Pavlakou, T., and Murray, I. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pp. 2338–2347, 2017.

Peng, K., Ping, W., Song, Z., and Zhao, K. Parallel neural text-to-speech. *arXiv preprint arXiv:1905.08459*, 2019.

Pharris, B. NV-WaveNet: Better speech synthesis using gpu-enabled WaveNet inference. In *NVIDIA Developer Blog*, 2018.

Ping, W., Peng, K., Gibiansky, A., Arik, S. O., Kannan, A., Narang, S., Raiman, J., and Miller, J. Deep Voice 3: Scaling text-to-speech with convolutional sequence learning. In *ICLR*, 2018.

Ping, W., Peng, K., and Chen, J. ClariNet: Parallel wave generation in end-to-end text-to-speech. In *ICLR*, 2019.

Prenger, R., Valle, R., and Catanzaro, B. WaveGlow: A flow-based generative network for speech synthesis. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.

Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Ren, Y., Ruan, Y., Tan, X., Qin, T., Zhao, S., Zhao, Z., and Liu, T.-Y. Fastspeech: Fast, robust and controllable text to speech. *arXiv preprint arXiv:1905.09263*, 2019.

Rezende, D. J. and Mohamed, S. Variational inference with normalizing flows. In *ICML*, 2015.

Ribeiro, F., Florêncio, D., Zhang, C., and Seltzer, M. CrowdMOS: An approach for crowdsourcing mean opinion score studies. In *ICASSP*, 2011.

Salimans, T. and Kingma, D. P. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 901–909, 2016.

Serrà, J., Pascual, S., and Segura, C. Blow: a single-scale hyperconditioned flow for non-parallel raw-audio voice conversion. *arXiv preprint arXiv:1906.00794*, 2019.

Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N., Yang, Z., Chen, Z., Zhang, Y., Wang, Y., Skerry-Ryan, R., et al. Natural TTS synthesis by conditioning WaveNet on mel spectrogram predictions. In *ICASSP*, 2018.

Sotelo, J., Mehri, S., Kumar, K., Santos, J. F., Kastner, K., Courville, A., and Bengio, Y. Char2wav: End-to-end speech synthesis. *ICLR workshop*, 2017.

Taigman, Y., Wolf, L., Polyak, A., and Nachmani, E. VoiceLoop: Voice fitting and synthesis via a phonological loop. In *ICLR*, 2018.

Tran, D., Vafa, K., Agrawal, K. K., Dinh, L., and Poole, B. Discrete flows: Invertible generative models of discrete data. *arXiv preprint arXiv:1905.10347*, 2019.

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. WaveNet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

Van den Oord, A., Kalchbrenner, N., Espeholt, L., Vinyals, O., Graves, A., et al. Conditional image generation with pixelcnn decoders. In *Advances in neural information processing systems*, pp. 4790–4798, 2016.

van den Oord, A., Li, Y., Babuschkin, I., Simonyan, K., Vinyals, O., Kavukcuoglu, K., Driessche, G. v. d., Lockhart, E., Cobo, L. C., Stimberg, F., et al. Parallel WaveNet: Fast high-fidelity speech synthesis. In *ICML*, 2018.

Wang, X., Takaki, S., and Yamagishi, J. Neural source-filter-based waveform model for statistical parametric speech synthesis. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5916–5920. IEEE, 2019.

Wang, Y., Skerry-Ryan, R., Stanton, D., Wu, Y., Weiss, R. J., Jaitly, N., Yang, Z., Xiao, Y., Chen, Z., Bengio, S., Le, Q., Agiomyrgiannakis, Y., Clark, R., and Saurous, R. A. Tacotron: Towards end-to-end speech synthesis. In *Interspeech*, 2017.

Yamamoto, R., Song, E., and Kim, J.-M. Parallel wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram. *arXiv preprint arXiv:1910.11480*, 2019a.

Yamamoto, R., Song, E., and Kim, J.-M. Probability density distillation with generative adversarial networks for high-quality parallel waveform generation. *arXiv preprint arXiv:1904.04472*, 2019b.

Yu, F. and Koltun, V. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

Appendix

A. Squeezing time-domain samples on channel dimension may raise artifacts

It is inefficient for modeling raw waveform by squeezing the time-domain samples on channel dimension and applying feed-forward transformation, because one can lose the temporal order information within the squeezed sequence. As a result of doing this, the synthesized audios from WaveGlow may contain constant frequency noise (see Figure 4). Note that, the frequencies of the noises (horizontal lines on spectrogram) are directly related to the squeezing size.

(a) squeezing size = 4. The frequency of noise is $\frac{22.05}{4}$ 5.5 kHz (the horizontal line).

(b) squeezing size = 8. The frequencies of the noises are $\frac{22.05}{4}$ = 5.5 kHz and $\frac{22.05}{8}$ 2.75 kHz.

Figure 4. (best viewed in color) The spectrograms of synthesized audios from WaveGlow by setting (a) squeezing size = 4 without early output, and (b) squeezing size = 8 with early output (default in the WaveGlow repository).