WaveNODE: A Continuous Normalizing Flow for Speech Synthesis

Hyeongju Kim [1] **Hyeonseung Lee** [1] **Woo Hyun Kang** [1] **Sung Jun Cheon** [1] **Byoung Jin Choi** [1] **Nam Soo Kim** [1]

Abstract

In recent years, various flow-based generative models have been proposed to generate high-fidelity waveforms in real-time. However, these models require either a well-trained teacher network or a number of flow steps making them memory-inefficient. In this paper, we propose a novel generative model called WaveNODE which exploits a continuous normalizing flow for speech synthesis. Unlike the conventional models, WaveNODE places no constraint on the function used for flow operation, thus allowing the usage of more flexible and complex functions. Moreover, WaveNODE can be optimized to maximize the likelihood without requiring any teacher network or auxiliary loss terms. We experimentally show that WaveNODE achieves comparable performance with fewer parameters compared to the conventional flow-based vocoders.

1. Introduction

Modern end-to-end speech synthesis models mostly consist of two stages: (1) transforming character embeddings to acoustic features such as mel-spectrograms (Ping et al., 2017; Shen et al., 2018; Vasquez & Lewis, 2019; Ren et al., 2019), and (2) synthesizing time-domain waveforms from the derived acoustic features (Oord et al., 2016; 2017; Ping et al., 2018; Kim et al., 2018; Prenger et al., 2019). In various end-to-end speech synthesis models, the WaveNet vocoder conditioned on mel-spectrograms is employed for the second stage to generate high-fidelity raw audio (Ping et al., 2017; Shen et al., 2018). However, samples cannot be obtained in real-time with the WaveNet vocoder due to its autoregressive nature. To enable fast sampling, flow-based generative models have recently attracted attention in the field of speech synthesis (Oord et al., 2017; Kim et al., 2018; Ping et al., 2018; Prenger et al., 2019).

[1]Department of Electrical and Computer Engineering and INMC, Seoul National University, South Korea. Correspondence to: Hyeongju Kim <hjkim@hi.snu.ac.kr>.

Second workshop on *Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models* (ICML 2020), Virtual Conference

In order to generate audio samples in real-time, parallel WaveNet (Oord et al., 2017) and ClariNet (Ping et al., 2018) employ inverse autoregressive flow (IAF) (Kingma et al., 2016) which takes advantage of the inverse transformation of an autoregressive function. Although IAF allows a parallel sampling procedure, it is not suitable to directly train the parallel WaveNet or ClariNet according to the maximum likelihood criterion. Instead, the parallel WaveNet and ClariNet are trained through probability density distillation which requires a well-trained teacher network. Additional hand-engineered objective functions are also needed to make the training procedure stable and to produce high quality audio.

FloWaveNet (Kim et al., 2018) and WaveGlow (Prenger et al., 2019) adopt an affine coupling layer which was originally proposed in Real NVP (Dinh et al., 2016). The affine coupling layer provides a simple inverse transformation and tractable determinant of the Jacobian. Unlike the IAF-based flow models, both the inference and sampling processes are parallelizable so that these models can be trained according to the maximum likelihood criterion without any auxiliary loss terms. However, Real NVP-based models require a number of flow steps to perform density estimation accurately as the affine coupling layer is too inflexible and simple. In this respect, FloWaveNet and WaveGlow are considered inherently memory-inefficient models.

Recently, Chen et al. (2018) introduced a new technique in which the hidden units are assumed to be continuously time-varying. In this framework, the continuous-time dynamics of the hidden units and their probability densities are described by deep neural networks. A continuous normalizing flow (CNF) is specified by these two dynamics using the instantaneous change of variables formula. Contrary to the discrete normalizing flows, the CNF does not impose any restrictions on its architecture and allows to use a quite flexible function for flow transformation as shown in Fig. 1. Here, in light of the advantages of the CNF, we propose a novel generative flow for speech synthesis called WaveNODE. WaveNODE generates high-fidelity waveforms from the corresponding mel-spectrograms with much fewer parameters compared to the conventional flow-based models by replacing the discrete normalizing flows with CNF.

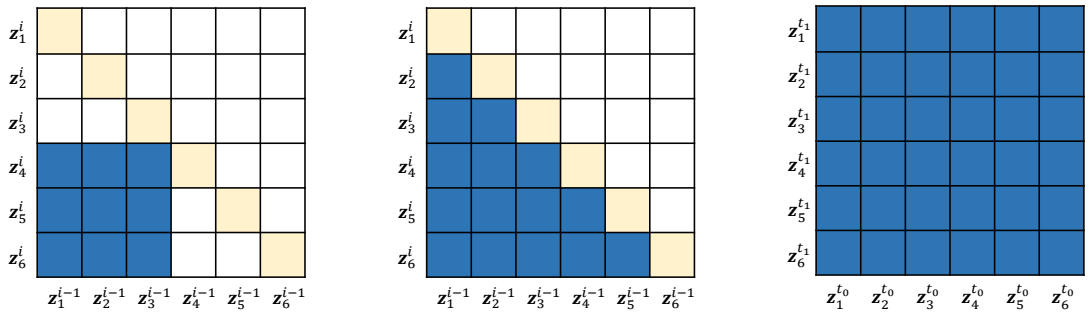

Figure 1. The Jacobian matrix of transformation in (a) Real NVP-based model, (b) IAF-based model, and (c) CNF-based model when the dimension of hidden state is 6. In (a) and (b), $\boldsymbol{z}_d^i$ stands for the d-th element of the i-th layer hidden state $\boldsymbol{z}^i$. In (c), $\boldsymbol{z}_d^t$ stands for the d-th element of time-varying variable $\boldsymbol{z}(t)$. The light-yellow cells represent the linear dependencies, the blue cells represent the non-linear dependencies and the white cells denote independent relations. The transformation in the CNF-based model is a lot more flexible than the others as the CNF does not impose any constraint on the type of functions used for flow transformation.

2. WaveNODE

We propose WaveNODE which takes advantage of the CNF for speech synthesis. WaveNODE is capable of generating high-fidelity waveforms from mel-spectrograms with a few flow steps. Moreover, WaveNODE does not require a teacher network or additional loss terms for training. The overall structure of the proposed WaveNODE is shown in Figure 2. We describe the CNF and the hierarchical architecture of WaveNODE in the next subsections.

2.1. Continuous Normalizing Flow

In Neural ODEs (Chen et al., 2018), the continuous dynamics of time-varying hidden units $\boldsymbol{z}(t) \in \mathbb{R}^D$ are parameterized using an ODE

$$\frac{d\boldsymbol{z}(t)}{dt} = \boldsymbol{f}(\boldsymbol{z}(t), t), \tag{1}$$

where $\boldsymbol{f}(\boldsymbol{z}(t), t)$ is implemented by a neural network. Here, t represents the time-step of solving the ODE, not the temporal axis of waveforms. Applying Eq. (1), the change in log-density of $\boldsymbol{z}(t)$ follows a differential equation given by

$$\frac{\partial \log p(\boldsymbol{z}(t))}{\partial t} = -\operatorname{Tr}\left(\frac{\partial \boldsymbol{f}(\boldsymbol{z}(t), t)}{\partial \boldsymbol{z}(t)}\right), \tag{2}$$

which is called the instantaneous change of variables formula. To avoid the $\mathcal{O}(D^2)$ cost of computing the trace operation, an unbiased estimate of Eq. (2) can be derived using the Hutchinson's trace estimator (Hutchinson, 1990) as follows:

$$\begin{aligned}\frac{\partial \log p(\boldsymbol{z}(t))}{\partial t} &= -\mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\boldsymbol{\epsilon}^\top \frac{\partial \boldsymbol{f}(\boldsymbol{z}(t), t)}{\partial \boldsymbol{z}(t)}\boldsymbol{\epsilon}\right] \\ &\approx -\frac{1}{K}\sum_{k=1}^{K}\left[\boldsymbol{\epsilon}_k^\top \frac{\partial \boldsymbol{f}(\boldsymbol{z}(t), t)}{\partial \boldsymbol{z}(t)}\boldsymbol{\epsilon}_k\right],\end{aligned} \tag{3}$$

where $\boldsymbol{\epsilon}_k \in \mathbb{R}^D$ is a noise vector drawn from $p(\boldsymbol{\epsilon})$ such that $\mathbb{E}[\boldsymbol{\epsilon}] = \mathbf{0}$ and $\mathrm{Cov}(\boldsymbol{\epsilon}) = \boldsymbol{I}$ (Grathwohl et al., 2018). Eq. (3) enables a fast computation of Eq. (2) since $\boldsymbol{\epsilon}_k^\top \frac{\partial \boldsymbol{f}(\boldsymbol{z}(t),t)}{\partial \boldsymbol{z}(t)}$ (vector-Jacobian product) can be efficiently calculated via deep learning libraries (e.g., PyTorch (Paszke et al., 2019)) without explicitly writing out the Jacobian matrix. By setting $K = 1$, we obtain an unbiased stochastic estimator of the dynamics of the log-likelihood with $\mathcal{O}(D)$ cost.

To train a generative model based on the CNF, we first assume that a latent variable $\boldsymbol{z}_0 \in \mathbb{R}^D$ follows a simple distribution $p_Z(\boldsymbol{z}_0)$. Next, let $\boldsymbol{f}(\boldsymbol{z}(t), t)$ be the dynamics of $\boldsymbol{z}(t)$ with the initial value $\boldsymbol{z}(t_0) = \boldsymbol{z}_0$. Given a datapoint $\boldsymbol{x} \in \mathbb{R}^D$, the corresponding latent variable $\boldsymbol{z}_0$ is obtained by solving the following ODE:

$$\boldsymbol{z}_0 = \boldsymbol{z}(t_0) = \int_{t_1}^{t_0} \boldsymbol{f}(\boldsymbol{z}(t), t)dt + \boldsymbol{z}(t_1), \tag{4}$$

where the final value $\boldsymbol{z}(t_1) = \boldsymbol{x}$. The log-likelihood of $\boldsymbol{x}$ can also be determined by solving another ODE

$$\log p_X(\boldsymbol{x}) = \log p_Z(\boldsymbol{z}_0) - \int_{t_0}^{t_1} \boldsymbol{\epsilon}^\top \frac{\partial \boldsymbol{f}(\boldsymbol{z}(t), t)}{\partial \boldsymbol{z}(t)}\boldsymbol{\epsilon} dt, \tag{5}$$

where $\boldsymbol{\epsilon}$ is a sampled noise vector from $p(\boldsymbol{\epsilon})$. Since $\boldsymbol{z}(t)$ varies along the vector field $\boldsymbol{f}(\boldsymbol{z}(t), t)$, the sampling process can be performed by simply reversing the time interval in Eq. (4) as follows:

$$\boldsymbol{x} = \int_{t_0}^{t_1} \boldsymbol{f}(\boldsymbol{z}(t), t)dt + \boldsymbol{z}_0. \tag{6}$$

Unlike the conventional normalizing flows, $\boldsymbol{f}(\boldsymbol{z}(t), t)$ is not required to be invertible or have a tractable Jacobian. Hence, any arbitrary functions can be employed for $\boldsymbol{f}(\boldsymbol{z}(t), t)$.

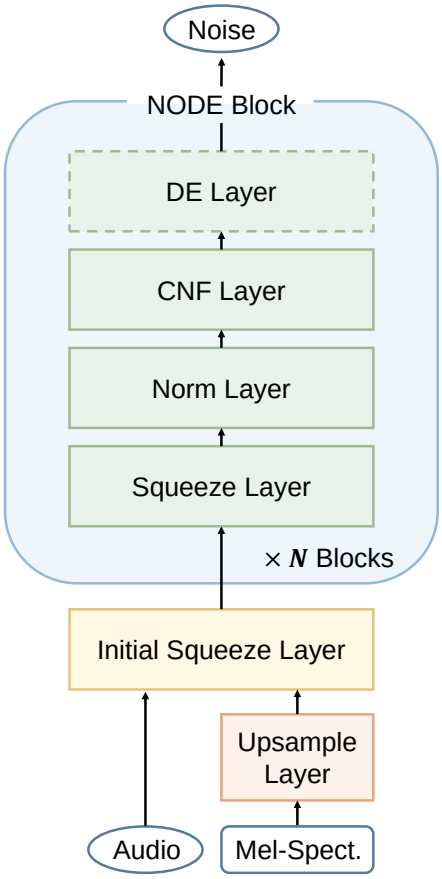

Figure 2. Overall structure of WaveNODE.

2.2. CNF Layer

Since we are interested in retrieving a time-domain signal from its mel-spectrogram, we apply a CNF framework to estimate the conditional distribution of audio samples. Given an upsampled mel-spectrogram $\boldsymbol{c}$ to full time resolution, Eq. (1) and Eq. (2) can be extended to a conditional formulation as follows:

$$\frac{d\boldsymbol{z}(t)}{dt} = \boldsymbol{f}_{CNF}(\boldsymbol{z}(t), t, \boldsymbol{c}), \tag{7}$$

$$\frac{\partial \log p(\boldsymbol{z}(t)|\boldsymbol{c})}{\partial t} = -\boldsymbol{\epsilon}^\top \frac{\partial \boldsymbol{f}_{CNF}(\boldsymbol{z}(t), t, \boldsymbol{c})}{\partial \boldsymbol{z}(t)} \boldsymbol{\epsilon}. \tag{8}$$

To capture the long-term dependency between audio samples, WaveNODE adopts a non-causal dilated convolutional network similar to the WaveGlow (Prenger et al., 2019) architecture for $\boldsymbol{f}_{CNF}(\boldsymbol{z}(t), t, \boldsymbol{c})$. Let V and W be a convolutional layer, and U be a linear projection. The activation function of the layers used for $\boldsymbol{f}_{CNF}(\boldsymbol{z}(t), t, \boldsymbol{c})$ is defined as

$$\boldsymbol{a}^f_{in} = W_f * \boldsymbol{z}(t) + V_f * \boldsymbol{c} + U_f t, \tag{9}$$

$$\boldsymbol{a}^g_{in} = W_g * \boldsymbol{z}(t) + V_g * \boldsymbol{c} + U_g t, \tag{10}$$

$$\boldsymbol{a}_{out} = \tanh(\boldsymbol{a}^f_{in}) \odot \text{ sigmoid}(\boldsymbol{a}^g_{in}), \tag{11}$$

where * denotes a convolution operator, super/subscripts f and g denote filter and gate, respectively. Note that WaveNODE uses t as a global condition via broadcasting and $\boldsymbol{c}$ as a local condition. The CNF layer receives the initial value $\boldsymbol{z}(t_0)$, time interval $[t_0, t_1]$ and the condition $\boldsymbol{c}$ as inputs. Using a black-box ODE solver, the CNF layer outputs the final value $\boldsymbol{z}(t_1)$ and the change in log-likelihood $\Delta \log p(\boldsymbol{z}(t))$.

2.3. Squeeze Layer

A squeeze layer rearranges an input tensor of shape $(C \times L)$ to form an output tensor of shape $(qC \times \frac{L}{q})$ where q is a scale factor. Increasing the number of channels by q times, the squeeze layer enlarges the receptive field of the convolutional networks linearly. This also helps each NODE block to focus on different temporal dependencies.

Since speech signals have a very high temporal resolution, it may not be desirable to directly use a mono audio signal $\boldsymbol{x}$ with the shape $(1 \times L)$ as input to a convolutional network. To deal with this, WaveNODE employs the initial squeeze layer to transform an input tensor of shape $(1 \times L)$ into an output tensor of shape $(q_{init} \times \frac{L}{q_{init}})$ given an initial scale factor q_{init}. This is the same as using Squeeze Layer with a scale factor $q \cdot q_{init}$ in the first NODE Block.

2.4. Norm Layer

WaveNODE incorporates a norm layer to alleviate the difficulties that arise when training deep neural networks. In this work, we consider two variants of batch normalization (Ioffe & Szegedy, 2015).

2.4.1. ActNorm

With trainable parameters $\boldsymbol{s}$ and $\boldsymbol{b}$ $(\boldsymbol{s}, \boldsymbol{b} \in \mathbb{R}^C)$, actnorm (Kingma & Dhariwal, 2018) performs per-channel affine transformation on $\boldsymbol{z}(t) \in \mathbb{R}^{C \times L}$

$$\boldsymbol{f}_{AN}(\boldsymbol{z}(t)_c) = \boldsymbol{s}_c \cdot \boldsymbol{z}(t)_c + \boldsymbol{b}_c, \tag{12}$$

where $\boldsymbol{s}_c$ and $\boldsymbol{b}_c$ are the c-th elements of $\boldsymbol{s}$ and $\boldsymbol{b}$, respectively, and $\boldsymbol{z}(t)_c \in \mathbb{R}^L$ is the c-th channel vector of $\boldsymbol{z}(t)$. The parameters $\boldsymbol{s}$ and $\boldsymbol{b}$ are initialized to normalize the pre-actnorm activations given an arbitrary initial batch (i.e., data dependent initialization). The change in log-likelihood obtained by passing $\boldsymbol{z}(t)$ through the actnorm layer can be computed as

$$\Delta \log p(\boldsymbol{z}(t)) = -L \sum_{c=1}^{C} \log |\boldsymbol{s}_c|. \tag{13}$$

2.4.2. Moving Batch Normalization

Moving batch normalization (MBN) exploits running averages instead of the current batch statistics as given by

$$\boldsymbol{f}_{MBN}(\boldsymbol{z}(t)_c) = \boldsymbol{s}_c \cdot \left(\frac{\boldsymbol{z}(t)_c - \boldsymbol{\mu}_c}{\boldsymbol{\sigma}_c} \right) + \boldsymbol{b}_c, \qquad (14)$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are the running averages of mean and standard deviation for each channel ($\boldsymbol{\mu}, \boldsymbol{\sigma} \in \mathbb{R}^C$), and subscript c denote the c-th channel component of variables. Similar to Eq. (13), the change in log-likelihood can be computed as

$$\Delta \log p(\boldsymbol{z}(t)) = -L \sum_{c=1}^{C} \left(\log |\boldsymbol{s}_c| - \log |\boldsymbol{\sigma}_c| \right). \qquad (15)$$

2.5. NODE Block

The NODE block is the primary component of WaveNODE, which basically consists of a squeeze layer, a norm layer, and a CNF layer as shown in Figure 2. Similar to FloWaveNet (Kim et al., 2018), WaveNODE stacks several NODE blocks and factors out half of the feature channels at a selected few NODE blocks. Factored-out channels are assumed to be Gaussian whose mean and variance are computed via a density estimation layer (DE layer) using the remaining channels as inputs. These statistics are used to evaluate the likelihood of the factored-out channels. The remaining channels are further passed through deeper NODE blocks and transformed into standard Gaussian noise in the end.

3. Experiments

In order to evaluate the performance of WaveNODE, we conducted a set of experiments using the LJ speech dataset (Ito, 2017) with the Griffin-Lim algorithm (Griffin & Lim, 1984) and various neural vocoders. All the neural vocoder models were trained for 7 days on a single NVIDIA RTX 2080Ti GPU. For the subjective evaluation of audio fidelity, we performed a 5-scale mean opinion score (MOS) test with 33 audio examples per model and 27 participants. The audio examples were randomly selected from the test set. Each participant listened to the audio examples played in random order and evaluated the audio quality. Confidence intervals of MOS were calculated using the method proposed in Ribeiro et al. (2011). To encourage reproducibility, we attach the code for WaveNODE and the audio samples used in the experiments [1] [2]. Also, we describe the configuration of other vocoders in Appendix A.

[1] https://github.com/ANLGBOY/WaveNODE/.
[2] https://wavenode-example.github.io/.

Table 1. Mean opinion scores (MOS) with 95% confidence intervals and conditional log-likelihoods (CLL) on the test set. The CLL obtained by solving neural ODEs is labeled with †.

Model	Number of parameters	CLL	MOS
Ground Truth	-	-	4.84±0.06
Griffin-Lim	-	-	2.82±0.26
WaveNet	4.8M	4.616	4.48±0.16
WaveGlow	87.9M	4.501	4.17±0.15
WaveGlow	17.1M	4.366	1.75±0.20
FloWaveNet	182.6M	4.449	2.99±0.19
FloWaveNet	18.6M	4.249	1.22±0.18
WaveNODE	16.2M	4.497†	3.53±0.18

3.1. WaveNODE

WaveNODE has 4 NODE blocks, each of which basically consists of a CNF layer, an actnorm layer, and a squeeze layer with scale factor $q = 2$. For the CNF layer, WaveNODE employs a 4-layer non-causal WaveNet with kernel size 3 where the channels of residual and skip connections are set to 128. Since WaveNODE stacks only a few NODE blocks, we set the base of dilation to 3 to increase the receptive field. At the end of the second NODE block, half of feature channels are factored out and their likelihood is estimated via a DE layer with a 2-layer network. The initial scale factor q_{init} is set to 4. For upsampling mel-spectrograms, a single layer of transposed 1D convolution is incorporated.

4. Results

4.1. Audio Fidelity and Conditional Log-Likelihood

We report the results of model comparison on a 5-scale MOS and conditional log-likelihoods (CLL) in Table 1. In both the MOS and the CLL tests, the performance of the WaveNet vocoder was the best among all vocoders used in the experiments. Among the flow-based vocoder models, WaveGlow gained the highest scores for both MOS and CLL of 4.17 and 4.501, respectively. The MOS score of WaveNODE was between the scores of WaveGlow and FloWaveNet. Note that WaveGlow and FloWaveNet have a much larger number of parameters than WaveNODE. To verify that only WaveNODE is capable of generating high-fidelity audio with a few flow steps, we also evaluated the performance of the compressed models of WaveGlow and FloWaveNet. As shown in Table 1, the compact models of WaveGlow and FloWaveNet received relatively poor MOS scores of 1.75 and 1.22, respectively. The results demonstrate that the ability of WaveNODE to use constraint-free functions for flow transformation allows the implementation of a memory-efficient vocoder that is capable of generating high-fidelity waveforms.

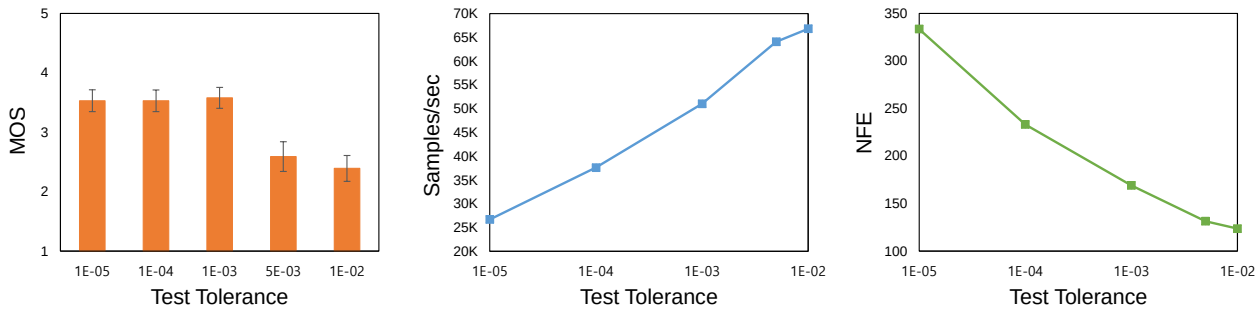

Figure 3. Mean opinion scores (MOS) with 95% confidence intervals (left), the number of samples generated per second (middle) and the number of function evaluations (NFE) (right) versus test tolerance. The results demonstrate that we can boost the sampling speed of WaveNODE by lowering the tolerance to some extent without significantly degrading audio fidelity.

Table 2. Comparison of synthesis speed. All models are benchmarked using a single RTX 2080Ti GPU.

MODEL	SAMPLES/SEC
WAVENET	56
WAVEGLOW	328,690
FLOWAVENET	320,062
WAVENODE	51,045

Table 3. Evaluations on mean opinion score (MOS) and conditional log-likelihood (CLL) for WaveNODE models with different types of norm layer.

MODEL	NORM LAYER	CLL	MOS
WAVENODE	ACTNORM	4.497	3.53±0.18
WAVENODE	MBN	4.460	3.36±0.17
WAVENODE	NONE	4.457	3.22±0.17

4.2. Synthesis Speed

WaveNODE is able to control the synthesis speed by tuning the accuracy of the black-box ODE solver. When solving an ODE, the ODE solver predicts the errors and adjusts its step-size to reduce the errors below a user set tolerance. To study the effect of changing the accuracy of the ODE solver, we tested the synthesis speed on a single RTX 2080Ti GPU. More specifically, we divided the total number of generated sample points by the total time, measured the number of function evaluations (NFE), and evaluated the audio quality by modifying the tolerance which had been fixed to 10^{-5} during training. The middle and right graphs in Figure 3 represent that the synthesis speed increased steadily by allowing higher tolerances at logarithmic scale. On the other hand, the audio quality was little affected until the tolerance was set to 10^{-3}. However, the audio quality dropped sharply after setting the tolerance to higher than 5×10^{-3}. The results suggest that WaveNODE can increase the sampling speed to some extent without seriously degrading the audio fidelity.

To compare the synthesis speed of the various neural vocoders, we counted the number of samples generated per second and report the results in Table 2. We set the test tolerance of WaveNODE to 10^{-3}. While WaveNet achieved the best result in the MOS test, it showed the worst performance on synthesis speed as it generates one sample point at a time (i.e., ancestral sampling). On the other hand, WaveNODE generated 51K samples/sec, which was a lot faster than WaveNet due to the parallel sampling process of flow operation. This suggests that WaveNODE is capable of generating audio samples in real-time even though WaveNODE has to solve complex ODEs in every flow operation. The sampling speeds of FloWaveNet and WaveGlow were the fastest since these models are not required to solve ODEs.

4.3. Type of Norm Layer

In Table 3, we report the performance of WaveNODE models with different norm layers. The results show that the norm layer is advantageous for WaveNODE to achieve good performance when adopting a multi-scale architecture unlike WaveGlow which can be fully trained without batch normalization. Popular CNF-based models often employ an MBN layer for stable training while constructing a deep architecture (Grathwohl et al., 2018; Yang et al., 2019). In our experiments, WaveNODE showed the finest results in terms of both MOS and CLL when employing an actnorm layer rather than the MBN layer. This implies that CNF-based models for speech data can be trained more efficiently by exploiting the actnorm layer.

4.4. Analysis of Training Progress

One of the major drawbacks shared by CNF-based models is the computational cost of the black-box ODE solver. The ODE solver creates a deep computational graph since it

Table 4. Training configurations of flow-based vocoder models.

MODEL	BATCH SIZE	EPOCH	ITERATION	CLL	MOS	TRAINING TIME
WAVEGLOW	8	240	354K	4.501	4.17±0.15	7 DAYS
FLOWAVENET	2	138	814K	4.449	2.99±0.19	7 DAYS
WAVENODE	20	46	27K	4.497	3.53±0.18	7 DAYS
WAVENODE	4	26	77K	4.452	2.91±0.17	7 DAYS

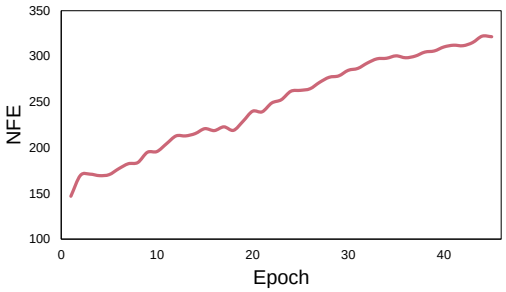

Figure 4. Evolution of the number of function evaluations (NFE) during training.

finds the solutions to complex ODEs specified by neural networks in an iterative way. Due to this large amount of computation, the mini-batches in the CNF-based models are processed for a long time. Table 4 shows that processing time per mini-batch in WaveNODE was significantly longer than the conventional flow-based models. While WaveGlow went through 240 epochs in 7 days, WaveNODE performed only 46 epochs during the same period. Interestingly, on the other hand, WaveNODE showed decent performance with less training steps. This implies that WaveNODE is trained more efficiently per mini-batch due to the flexible functions used for flow transformation. We also tested whether reducing the batch size to increase the number of iterations can improve the performance of WaveNODE, but the resultant performance was degraded as shown in Table 4.

It has been reported that the NFE in CNF-based models increases as training progresses (Chen et al., 2018; Grathwohl et al., 2018). We also observed a similar phenomenon when training WaveNODE and report the overall trend of the NFE consumed for inference in Figure 4. The main reason for this phenomenon is that ODEs become more complicated to accurately estimate the conditional distribution of waveforms. Since NFE directly affects the time taken to process a mini-batch, we plan to research how to prevent an increase of NFE in future work.

4.5. Analysis of Dilation

WaveNODE is composed of only a few flow steps unlike the conventional flow-based models, which might result in the small receptive field. In order to capture the long-range temporal dependencies in audio signals, we basically set the base of dilation to 3 in WaveNODE for the previous experiments. To verify the effect of dilation, we trained the WaveNODE model with dilation of 2^i at the i-th layer and evaluated performance in terms of MOS and CLL. Indeed, the dilation was critical to the quality of audio samples generated by WaveNODE as shown in Table 5. We found that WaveNODE produces a trembling sound when the dilation is a multiple of 2 due to the narrow receptive field.

Table 5. Comparison of WaveNODE models with different dilations.

MODEL	DILATION (AT i-TH LAYER)	CLL	MOS
WAVENODE	3^i	4.497	3.53±0.18
WAVENODE	2^i	4.408	3.17±0.17

5. Conclusion

In this work, we presented the novel generative model, namely WaveNODE, which leverages a CNF for speech synthesis. We successfully applied the CNF framework to a large-dimensional data (e.g., audio) without any additional loss term. In the experiments, we demonstrated that WaveNODE shows comparable performance with fewer flow steps compared to the conventional flow-based models. Also, we verified that WaveNODE is able to synthesize audio samples in real-time due to the parallel sampling process. We believe that applying CNF to speech synthesis can be further developed and refined to produce more realistic waveforms.

Acknowledgments

This work was supported by Samsung Research Funding Center of Samsung Electronics under Project Number SRFC-IT1701-04.

References

Chen, T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. In *Advances in neural information processing systems*, pp. 6571–6583,

2018.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

Grathwohl, W., Chen, R. T., Betterncourt, J., Sutskever, I., and Duvenaud, D. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

Griffin, D. and Lim, J. Signal estimation from modified short-time fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(2):236–243, 1984.

Hutchinson, M. F. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 19(2):433–450, 1990.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Ito, K. The lj speech dataset. https://keithito.com/LJ-Speech-Dataset/, 2017.

Kim, S., Lee, S.-g., Song, J., Kim, J., and Yoon, S. Flowavenet: A generative flow for raw audio. *arXiv preprint arXiv:1811.02155*, 2018.

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. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pp. 4743–4751, 2016.

Oord, A. v. d., 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.

Oord, A. v. d., 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. *arXiv preprint arXiv:1711.10433*, 2017.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pp. 8024–8035, 2019.

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. *arXiv preprint arXiv:1710.07654*, 2017.

Ping, W., Peng, K., and Chen, J. Clarinet: Parallel wave generation in end-to-end text-to-speech. *arXiv preprint arXiv:1807.07281*, 2018.

Prenger, R., Valle, R., and Catanzaro, B. Waveglow: A flow-based generative network for speech synthesis. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3617–3621. IEEE, 2019.

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.

Ribeiro, F., Florêncio, D., Zhang, C., and Seltzer, M. Crowdmos: An approach for crowdsourcing mean opinion score studies. In *2011 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 2416–2419. IEEE, 2011.

Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N., Yang, Z., Chen, Z., Zhang, Y., Wang, Y., Skerrv-Ryan, R., et al. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4779–4783. IEEE, 2018.

Vasquez, S. and Lewis, M. Melnet: A generative model for audio in the frequency domain. *arXiv preprint arXiv:1906.01083*, 2019.

Yang, G., Huang, X., Hao, Z., Liu, M.-Y., Belongie, S., and Hariharan, B. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4541–4550, 2019.

A. Model Configuration

A.1. Griffin-Lim

The Griffin-Lim algorithm (Griffin & Lim, 1984) estimates the signal from its modified short-time Fourier transform (STFT) magnitude in an iterative way. For the experiments, we first approximated the STFT magnitude from the mel-spectrogram and applied the Griffin-Lim algorithm with 32 iterations for time-domain conversion.

A.2. WaveNet

We trained an autoregressive WaveNet whose output is a single Gaussian distribution. It has been shown that a single Gaussian WaveNet is capable of modeling raw waveforms without degradation compared to WaveNet models with mixture distribution (Ping et al., 2018). We stacked 2 dilated residual blocks of 10 layers with kernel size 2 and set the number of hidden units in both residual and skip connections to 128. To upsample the mel-spectrograms from frame-level to sample-level resolution, we employed two layers of transposed 2D convolution with one leaky ReLU activation.

A.3. FloWaveNet

FloWaveNet (Kim et al., 2018) consists of 8 context blocks, each of which contains 6 flow operations. For the affine coupling layers, FloWaveNet employs a 2-layer non-causal WaveNet with kernel size 3. FloWaveNet uses 256 channels for residual and skip connections. Also, FloWaveNet factors out half of the feature channels after 4 context blocks and uses another 2-layer network to estimate the distribution of factored-out channels. FloWaveNet incorporates the same upsampling module described in Section A.2.

For the experiments, we trained the compact version of FloWaveNet as well as the original model. We used 4 context blocks with 3 flow operations and factored out half of the feature channels after 2 context blocks for the compact model.

A.4. WaveGlow

WaveGlow (Prenger et al., 2019) is composed of 12 blocks, each of which contains an affine coupling layer and an invertible 1x1 convolution. WaveGlow employs 8-layer non-causal WaveNet networks for the affine coupling layers and one layer of transposed 1D convolution for upsampling mel-spectrograms. WaveGlow factors out 2 of feature channels after every 4 blocks.

In addition to the original WaveGlow network, we also trained the compressed model for the experiments. We stacked 9 blocks, used 4-layer networks, and factored out 2 of feature channels after every 3 blocks for the compressed model.