Interspeech 2022
18-22 September 2022, Incheon, Korea

SiD-WaveFlow: A Low-Resource Vocoder Independent of Prior Knowledge

Yuhan Li, Ying Shen, Dongqing Wang, Lin Zhang*

School of Software Engineering, Tongji University, P.R.China
{2031551,yingshen,wangdongqing,cslinzhang}@tongji.edu.cn

Abstract

Flow-based neural vocoders have demonstrated their effectiveness in generating high-fidelity speech in real-time. However, most flow-based vocoders are computationally heavy models which rely on large amounts of speech for model training. Witnessing the limitations of these vocoders, a new flow-based vocoder, namely Semi-inverse Dynamic WaveFlow (SiD-WaveFlow), for low-resource speech synthesis is proposed. SiD-WaveFlow can generate high-quality speech in real-time with the constraint of limited training data. Specifically, in SiD-WaveFlow, a module named Semi-inverse Dynamic Transformation (SiDT) is proposed to improve the synthesis quality as well as the computational efficiency by replacing the affine coupling layers (ACL) used in WaveGlow. In addition, a pre-emphasis operation is introduced to the training process of SiD-WaveFlow to further improve the quality of the synthesized speech. Experimental results have corroborated that SiD-WaveFlow can generate speech with better quality compared with its counterparts. Particularly, the TTS system integrating SiD-WaveFlow vocoder achieves 3.416 and 2.968 mean MOS on CSMSC and LJ speech dataset, respectively. Besides, SiD-WaveFlow converges much faster than WaveGlow at the training stage. Last but not least, SiD-WaveFlow is a lightweight model and can generate speech on edge devices with a much faster inference speed. The source code and demos are available at https://slptongji.github.io/.

Index Terms: speech synthesis, generative models, low-resource, neural vocoder

1. Introduction

A vocoder is an essential component in a speech synthetic system, which is used to generate speech waveforms from acoustic features (such as Mel-spectrograms). Neural vocoders have proven to outperform traditional source-filter vocoders [1, 2] for synthesizing high-quality speeches. The first groundbreaking neural vocoder was WaveNet [3], and soon many other approaches, such as WaveRNN [4], WaveFlow [5] and WaveGlow [6], etc., were developed. These vocoders based on generative networks offer state-of-the-art speech synthesis quality.

Existing generative models can be categorized into two types: the autoregressive ones that generate speech in sequence and the non-autoregressive ones that generate speech in parallel. Compared with autoregressive models [3, 7], those non-autoregressive ones, which include flow-based [5,6], Gan-based [8–10] and diffusion-based [11,12] models, consume much less inference time. However, due to structural complexity, these generative models rely on large amounts of data for training [13]. Therefore, it is a challenging problem to build a high-quality generative neural vocoder with limited training data.

Flow-based neural vocoders [5, 6] are a kind of non-autoregressive generative vocoder. They transform a probability density with a sequence of invertible mappings [14]. The flow-based models can be futher divided into two categories: 1) models based on autoregressive transforms (e.g. inverse autoregressive flow used in ClariNet [15]), and 2) models based on bipartite transforms (e.g. Glow [16] used in SqueezeWave [17] and WaveFlow [5]). Compared with autoregressive transforms, bipartite transforms have much simpler training pipelines but are usually more complex in model settings (e.g. deeper layers, more hidden nodes) [18]. Consequently, bipartite transform-based models require more data for sufficient model training. In addition to their model complexity, flow-based vocoders are computationally intensive, which makes them difficult to be deployed on the devices with limited computing resources, such as mobile phones, Raspberry Pi, etc.

Witnessing the above shortcomings of the existing flow-based models, a new flow-based model, namely SiD-WaveFlow, is proposed in this paper. SiD-WaveFlow can generate high-quality speech in low-resource conditions. Specifically, SiD-WaveFlow requires only five minutes of speech for model training without additional information. Besides, it has much fewer parameters compared with WaveGlow and its computational cost has been largely reduced. As a result, the inference speed of SiD-WaveFlow is much faster than WaveGlow. In addition, in the training stage, SiD-WaveFlow converges much faster.

Our contributions can be summarized as follows:

- A new flow-based neural vocoder SiD-WaveFlow is proposed for speech synthesis under low-resource conditions. It simply relies on 5-minute speeches for model training without external knowledge.
- In SiD-WaveFlow, a semi-inverse dynamic transformation (SiDT) layer is proposed to substitute the traditional affine coupling layers (ACL) used in bipartite transforms such as WaveGlow. Compared with ACL, the SiDT layer is more computationally efficient and converges much faster.
- A Prenet module which contains a pre-emphasis and a squeeze operation is added in SiD-WaveFlow, which helps to improve the quality of synthesized speech.

The experimental results have demonstrated the computational efficiency and effectiveness of the SiD-WaveFlow in low-resource speech synthesis.

2. Methodology

2.1. Preliminaries

A normalizing flow transforms a normalized density distribution to a target normalized density distribution through a sequence of invertible bijective transformation functions. Therefore, flow-based vocoders can generate speech $\boldsymbol{y}$ by applying a series of invertible deep neural networks $\boldsymbol{g}_i$ on the input Mel-spectrograms $\boldsymbol{x}$ using transformation $\boldsymbol{y} = \boldsymbol{g}_1 \circ \boldsymbol{g}_2 \circ \cdots \boldsymbol{g}_k(\boldsymbol{x})$. Because $\boldsymbol{g}_i$ is invertible, based on *change of variable rule*, the model can be trained by minimizing the negative log-likelihood

∗ Corresponding author.

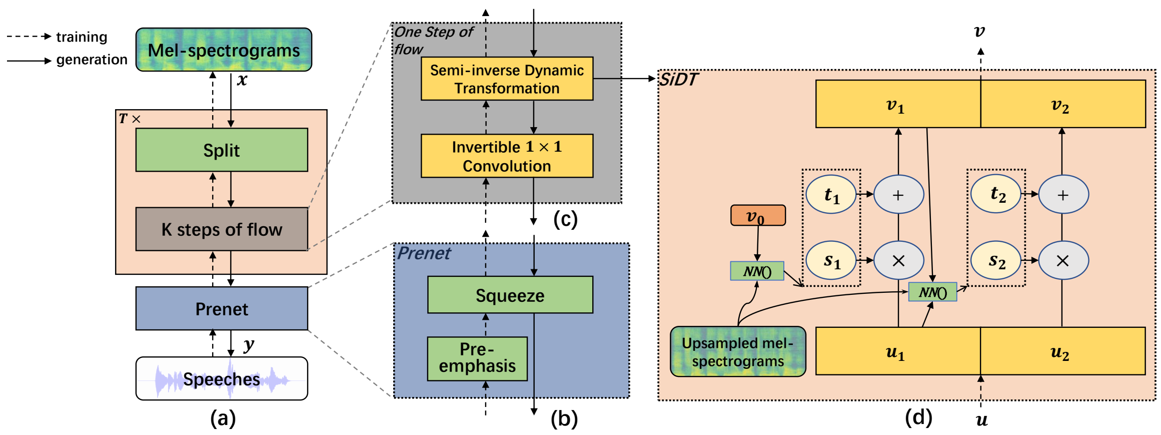

Figure 1: *The multi-scale architecture of SiD-WaveFlow. (a) SiD-WaveFlow transforms speeches to Mel-spectrograms at the training stage (dotted lines) and generates speeches from Mel-spectrograms (solid lines) at the speech generation stage; (b) The components in Prenet module; (c) The components in one step of flow; (d) The data flow in SiDT during the model training.*

of data as,

$$\log p(\boldsymbol{y}) = \log p(\boldsymbol{x}) + \sum_{i=1}^{K} \log \left| \det \frac{d\boldsymbol{g}_i^{-1}}{d\boldsymbol{g}_{i-1}^{-1}} \right| \tag{1}$$

where K is the number of invertible mappings, det $\frac{d\boldsymbol{g}_i^{-1}}{d\boldsymbol{g}_{i-1}^{-1}}$ is the Jacobian determinant of $\boldsymbol{g}_i^{-1}$, and $\boldsymbol{g}_0^{-1}(\boldsymbol{y})$ equals $\boldsymbol{y}$. Since $\boldsymbol{g}_i$ is invertible, $\boldsymbol{x}$ can be inferred by $\boldsymbol{x} = \boldsymbol{g}_K^{-1} \circ \boldsymbol{g}_{K-1}^{-1} \circ \cdots \boldsymbol{g}_1^{-1}(\boldsymbol{y})$.

WaveGlow [6] adopts the flow structure [16] to implement the above invertible transforms. It consists of a *squeeze operation*, an *invertible 1 × 1 convolution module*, and an *ACL module*. The invertible 1×1 convolution module and the ACL module form a *flow step* which will be repeated during the model training and the speech generation. These modules work in sequence to transform Mel-spectrograms to speech waveforms and vice versa.

2.2. Overview of SiD-WaveFlow

SiD-WaveFlow is a novel flow-based vocoder which is also based on generative networks. Its architecture has been shown in Figure 1 (a). Similar to WaveGlow, SiD-WaveFlow consists of a *Prenet module*, a *flow step* repeated K times, and a *split operation*. Particularly, K steps of flow and the split operation consist a T-scale architecture to accelerate the training speed. In the training stage, the speech samples are firstly fed into the Prenet module. Then, these preprocessed speech samples are passed to the flow step which will be repeated for K times. The flow step consists of two invertible transformations: an invertible 1×1 convolution layer and a newly proposed SiDT layer. In the end, the output of K steps of flow is split out to compute the loss.

2.3. Prenet

In Prenet module, two operations are performed on the input speeches, i.e. pre-emphasis and squeeze, as shown in Figure 1(b). Firstly, the original speeches are pre-emphasized to increase the high-frequency resolution of speeches as,

$$x'(n) = x(n) - ax(n-1) \tag{2}$$

where $x(n)$ and $x'(n)$ represent the n-th sampling point before and after pre-emphasis respectively and a is the pre-emphasis coefficient the value of which is between 0.9 and 1. After that, the squeeze operation which is used in WaveGlow is performed on the speech to increase the number of channels for further parallel computing. Specifically, the pre-emphasized speech sampling vectors are reshaped from $N \times 1$ to $\frac{N}{8} \times 8$ according to [19], where N is the number of sampling points in the speech.

2.4. Semi-inverse dynamic transformation

ACL [6] which is composed of a series of invertible neural networks is adopted by WaveGlow to implement a normalizing flow. However, ACL suffers from its limited computational performance. Inspired by the technique of inverse dynamic linear transformations (IDLT) [20], SiDT is proposed to replace ACL in the flow step. SiDT inherits the computational efficiency of ACL and the high performance of IDLT. The data flow in SiDT is shown in Figure 1(d) and is defined using Eqs. (3)-(8),

$$(\boldsymbol{u}_1, \boldsymbol{u}_2) = split(\boldsymbol{u}), \boldsymbol{v}_0 = \boldsymbol{0} \tag{3}$$
$$(\boldsymbol{s}_1, \boldsymbol{t}_1) = NN(\boldsymbol{m}(\boldsymbol{v}_0), \boldsymbol{h}) \tag{4}$$
$$\boldsymbol{v}_1 = \boldsymbol{s}_1 \odot \boldsymbol{u}_1 + \boldsymbol{t}_1 \tag{5}$$
$$(\boldsymbol{s}_2, \boldsymbol{t}_2) = NN(\boldsymbol{m}(\boldsymbol{v}_1, \boldsymbol{u}_1), \boldsymbol{h}) \tag{6}$$
$$\boldsymbol{v}_2 = \boldsymbol{s}_2 \odot \boldsymbol{u}_2 + \boldsymbol{t}_2 \tag{7}$$
$$\boldsymbol{v} = concat(\boldsymbol{v}_1, \boldsymbol{v}_2) \tag{8}$$

where $\boldsymbol{u}$ is the input vector, $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ are two vectors obtained by equally dividing vector $\boldsymbol{u}$, $\boldsymbol{v}_0$ is the vector $\boldsymbol{0}$, and $\boldsymbol{h}$ is the upsampled Mel-spectrograms of input speeches, which is used to adjust the output of SiDT. $\boldsymbol{m}$ can be any transformation such as addition or multiplication, and $NN(\cdot)$ is a modified gated convolution layer [21].$\boldsymbol{s}_1$, $\boldsymbol{s}_2$, $\boldsymbol{t}_1$, and $\boldsymbol{t}_2$ are the affine factor vectors computed by $NN(\cdot)$. Vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ are computed from $\boldsymbol{s}_1$, $\boldsymbol{s}_2$, $\boldsymbol{t}_1$, and $\boldsymbol{t}_2$ using Eqs. (5) and (7). The output vector $\boldsymbol{v}$ is finally obtained by horizontally concatenating $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$.

Since $\boldsymbol{v}_0$ is added intentionally as the extra input, unlike IDLT, it is initialized to the vector $\boldsymbol{0}$ instead of $\boldsymbol{1}$ to avoid possible noise interference. Moreover, the number of intermediate channels in $NN(\cdot)$ is cut down to alleviate the impact of the information bottleneck [17]. Specifically, $\boldsymbol{m}(x, y)$ is defined as $\boldsymbol{m}(x, y) = x + y$ to accelerate calculation and reduce the number of parameters. The training loss of SiDT can be easily computed as,

$$\log \left| \det \frac{d\boldsymbol{g}_{\mathrm{SiDT}}^{-1}(\boldsymbol{u})}{d\boldsymbol{u}} \right| = \log |\boldsymbol{s}_1| + \log |\boldsymbol{s}_2| \tag{9}$$

2.5. Invertible 1×1 convolution layer

In flow-based generative networks [6,16,19], an invertible 1×1 convolution is adopted to reorganize the order of the channels. Therefore, the same invertible 1×1 convolution layer is added before the SiDT layer to reorganize the order of intermediate variables in SiD-WaveFlow (see Figure 1(c)). To ensure the invertibility of the network, the weights of the convolution W are initialized with a random orthogonal matrix. The data transformation in an invertible 1×1 convolution layer during training can be defined as $\boldsymbol{g}_{1\times1conv}^{-1} = W\boldsymbol{z}$, where $\boldsymbol{z}$ is the vector either from the output of Prenet module or from the output of SiDT in the former flow step. The training loss of the invertible 1×1 convolution can be computed as,

$$\log\left|\det\frac{d\boldsymbol{g}_{1\times1conv}^{-1}(\boldsymbol{z})}{d\boldsymbol{z}}\right| = \log|\det W| \tag{10}$$

where $\det\frac{d\boldsymbol{g}_{1\times1conv}^{-1}(\boldsymbol{z})}{d\boldsymbol{z}}$ is the determinant of the Jacobian of the invertible 1×1 convolution.

2.6. Multi-scale architecture

The multi-scale architecture can extract information in different time scales and accelerate the training speed for a flow-based model [16]. As shown in Figure 1(a), SiD-WaveFlow's architecture contains T-scales. Each scale contains K steps of flow in SiD-WaveFlow. In the training stage, the first two dimensions of the vector are split out to compute the loss at the end of each scale. The remaining dimensions are input into the next scale. After T scales, all dimensions are concatenated together to form the output of SiD-WaveFlow. The operations in the multi-scale structure are defined by Eqs. (11)-(13),

$$\boldsymbol{h}_0 = \boldsymbol{r} \tag{11}$$

$$(\boldsymbol{x}_i, \boldsymbol{h}_i) = split(Flow(\boldsymbol{h}_{i-1})) \tag{12}$$

$$\boldsymbol{x} = concat(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots \boldsymbol{x}_T) \tag{13}$$

where $\boldsymbol{r}$ is the output vector of the Prenet module, $Flow()$ represents the transformation of K steps of flow, $split()$ represents the split operation which extracts $\boldsymbol{x}_i$ from the output of K steps of flow, $\boldsymbol{h}_i$ is the intermediate vector in SiD-WaveFlow, and $\boldsymbol{x}$ is the output Mel-spectrograms after training. Moreover, $\boldsymbol{h}_0$ is initialized with $\boldsymbol{r}$.

3. Experiments

3.1. Experiment setup

To validate the performance of SiD-WaveFlow, extensive experiments are performed on a workstation equipped with a GTX 2080Ti and a Raspberry Pi 4B. Two datasets are used for performance evaluation. One is Chinese Standard Mandarin Speech Corpus (CSMSC) [22]. It contains 10,000 periods of recordings read by a young woman which last about 12 hours. The other one is LJ speech dataset [23], which contains 13,100 English sentence fragments read by a female speaker. The recordings in LJ speech dataset last about 24 hours.

To simulate a low-resource condition for speech synthesis, only 5 minutes of speech is used to train the model. Taking the CSMSC dataset as an example, the training sets are constructed as follows. At first, half of the dataset, i.e. 5,000 speech samples, are used for the test. Then, the remaining samples are randomly divided into 45 non-overlapping training sets. Each training set contains about 90 samples with a total duration of about 5 minutes. Because there are 45 non-overlapping training sets, the evaluation experiment is conducted 45 times. In each round of evaluation, the model is trained using only one training set and generates 5,000 speeches based on the Mel-spectrograms extracted from testing samples. The overall performance of the model is the average quality of all the generated speeches obtained in 45 rounds of evaluation experiments. Similarly, for the LJ speech dataset, 5,000 samples are randomly selected and regarded as the testing samples. 45 non-overlapping training sets are constructed from the remaining samples and each training set contains 50 samples with a 5-minutes duration. Table 1 gives the configurations of the training and testing sets for the experiments.

Table 1: *The configurations of training and testing sets*

	CSMSC	LJ speech dataset
♯ of Testing samples	5,000	5,000
♯ of Training samples	90	50
Training samples duration	5min	5min
Round of evaluation	45	45

In order to compute the training loss, in the training stage of SiD-WaveFlow, Mel-spectrograms are extracted from the training samples. The dimension of Mel-spectrograms D_m is set to 80. The value of pre-emphasis coefficient a is set to 0.95. For the multi-scale architecture, the number of scales T is set to 3. Four steps of flow are adopted in each scale, i.e. $K = 4$. The number of channels in $NN(\cdot)$ is set to 128. Besides, the batch size is set to 6 according to hardware conditions. The learning rate l_r is initialized to $4e^{-4}$. Then an adaptive strategy is adopted to adjust the learning rate during the back-propagation.

3.2. Evaluation

In addition to SiD-WaveFlow, the model based on Griffin-Lim algorithm [1] and two flow-based models, which include WaveGlow and Wave-IDLT proposed by us, are evaluated for performance comparison. Wave-IDLT is a new flow-based model by simply replacing ACL with IDLT layers [20] in WaveGlow.

Speech quality is often measured by Mean Opinion Score (MOS). In the previous practice of evaluation, several human testers were invited to give MOS values for a set of randomly selected test speeches. However, such tests may lead to an evaluation bias because the number of speeches which can be evaluated by people is much smaller than the number of testing samples to be evaluated in our experiments. Instead, in this work, MOS values are automatically computed using MOSNet for each generated speech and an average MOS value is summarized over all the generated speeches. MOSNet [24] is an algorithm which can automatically predict the MOS of the speech and has been widely used for speech quality assessment [25,26].

3.2.1. Qualitative evaluation

Speech quality. In each round of the experiment, a training set is used to train three models respectively and each model generates 5,000 speeches based on the Mel-spectrograms extracted from the testing samples. The performance of four models, i.e. SiD-WaveFlow, Griffin-Lim, WaveGlow, and Wave-IDLT, are evaluated by the average MOS value on 5,000×45 generated speech samples. Specifically, Griffin-Lim serves as the baseline model for performance evaluation. The average MOS values corresponding to the four models have been listed in Table 2. In Table 2, SiD-WaveFlow offers the highest speech quality with average MOS values 3.416 and 2.968 on CSMSC and LJ speech dataset, respectively. The average MOS values of WaveGlow and Griffin-Lim are worse than SiD-WaveFlow on both datasets. Wave-IDLT has the worst performance with average MOS values only 2.785 on both datasets. The results indicate that SiDT combines the advantages of both ACL and IDLT and SiD-WaveFlow offers the best synthesis quality compared with its counterparts.

Table 2: *Average MOS of synthesized speeches over 45 rounds of evaluation*

	CSMSC	LJ speech dataset
Ground Truth	3.754±0.007	3.067±0.002
Griffin-Lim	3.146±0.009	2.882±0.005
WaveGlow	3.324±0.001	2.909±0.001
Wave-IDLT	2.785±0.001	2.785±0.001
SiD-WaveFlow	**3.416±0.001**	**2.968±0.001**

Convergence speed. The convergence speeds of three flow-based methods are compared on a workstation equipped with a GTX 2080Ti. The results are shown in Figure 2. For a better observation, the loss curve is smoothed with a weight of 0.95. Then the smoothed loss is resampled every 100 steps to estimate the convergence. The model is regarded to be convergent when the difference between adjacent sampling points is in $[0, -0.1]$. As shown in Table 3, SiD-WaveFlow behaves similarly to Wave-IDLT, each of which requires 3,700 and 3,826 steps to converge respectively. By contrast, WaveGlow converges after 7,778 steps. The fast convergence rate of SiD-WaveFlow benefits from the computational efficiency of SiDT.

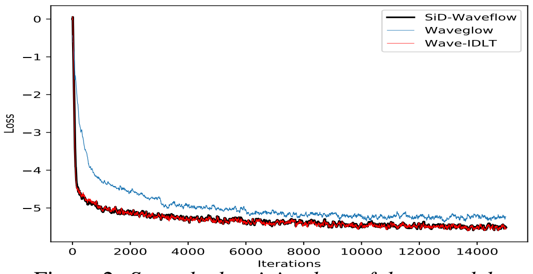

Figure 2: *Smoothed training loss of three models*

Table 3: *The number of steps required for convergence*

Models	Steps
WaveGlow	7,778
Wave-IDLT	3,826
SiD-WaveFlow	**3,700**

3.2.2. *Quantitative evaluation*

Inference speed. Three flow-based models are deployed to a workstation equipped with a GTX 2080Ti and a Raspberry Pi 4B to measure the inference speeds. The results are listed in Table 4. SiD-WaveFlow can generate 522k and 5.1k samples per second on the workstation and Raspberry Pi 4B respectively, which is much faster than the other two methods. In particular, Wave-IDLT fails to run on the Raspberry Pi 4B due to limited computation resources. The performance of SiD-WaveFlow on the workstation fully meets the requirement of real-time generation of high-quality speech (48kHz). Its fast inference speed benefits from its lighter structure which only contains 63.1M parameters. As a comparison, WaveGlow and Wave-IDLT contains 87.9M and 169.2M parameters, respectively.

Table 4: *The number of samples generated per second*

Models	Workstation	Raspberry Pi	Parameter#
WaveGlow	405k	4.4k	87.9M
Wave-IDLT	139k	failed	169.2M
SiD-WaveFlow	**522k**	**5.1k**	**63.1M**

3.2.3. *Ablation study*

To demonstrate the necessity of the pre-emphasis operation adopted by SiD-WaveFlow, the quality of speeches generated by SiD-WaveFlow with and without pre-emphasis are compared. Respectively, 20 speeches are randomly selected from the testing set generated by the two models. These speeches are mixed with the original speeches and each speech is evaluated by 10 testers. The average MOS values for two models are shown in Table 5 with 95% confidence intervals. It can be seen that the average MOS of the speeches generated by SiD-WaveFlow is about 3.62 which is much higher than that of the speeches generated by the model without using pre-emphasis operation. The results demonstrate that the pre-emphasis operation can improve the quality of the synthesized speech.

Table 5: *The Average MOS of speeches generated by SiD-WaveFlow and the model without pre-emphasis operation*

Model	MOS
Ground Truth	3.925 ± 0.008
SiD-WaveFlow without pre-emphasis	3.55±0.02
SiD-WaveFlow (with pre-emphasis)	**3.62±0.019**

4. Conclusion

The vocoders, which transform acoustic features into speech waveforms, play an important role in modern TTS systems. Low-resource conditions in which training data is insufficient raise a great challenge for the existing neural vocoders. Many existing solutions resort to pretraining models and then fine-tuning the models on the specific task using limited training data. By contrast, in this paper, a new flow-based vocoder, namely SiD-WaveFlow, is proposed for low-resource speech synthesis without pretraining techniques.

SiD-WaveFlow is the first flow-based vocoder working under extremely low-resource conditions. It can be well trained using only 5-minute speeches without external knowledge. Specifically, in SiD-WaveFlow, a new transformation named SiDT is proposed to reduce the computational cost raised by ACL and improve the speech synthesis quality simultaneously. Besides, a Prenet module is added into the architecture of SiD-WaveFlow to further improve the speech synthesis quality. The effectiveness and efficiency of SiD-WaveFlow for speech synthesis in low-resource conditions compared with its competitors has been corroborated by extensive experiments.

5. Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grant 61972285, in part by the Natural Science Foundation of Shanghai under Grant 19ZR1461300, in part by the Shanghai Municipal Science and Technology Major Project under Grant 2021SHZDZX0100, and in part by the Fundamental Research Funds for the Central Universities.

6. References

[1] D. Griffin and J. Lim, "Signal estimation from modified short-time fourier transform," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 1984, pp. 236–243.

[2] M. Morise, F. Yokomori, and K. Ozawa, "World: A vocoder-based high-quality speech synthesis system for real-time applications," *IEICE Transactions on Information and Systems*, vol. 99, no. 7, pp. 1877–1884, 2016.

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

[4] N. Kalchbrenner, E. Elsen, K. Simonyan, S. Noury, N. Casagrande, E. Lockhart, F. Stimberg, A. van den Oord, S. Dieleman, and K. Kavukcuoglu, "Efficient neural audio synthesis," 2018, pp. 2410–2419.

[5] W. Ping, K. Peng, K. Zhao, and Z. Song, "WaveFlow: A compact flow-based model for raw audio," in *International Conference on Machine Learning*, 2020, pp. 7706–7716.

[6] R. Prenger, R. Valle, and B. Catanzaro, "WaveGlow: A flow-based generative network for speech synthesis," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2019, pp. 3617–3621.

[7] A. van den Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. van den Driessche, E. Lockhart, L. Cobo, and F. Stimberg, "Parallel waveNet: Fast high-fidelity speech synthesis," in *International Conference on Machine Learning*, 2018, pp. 3918–3926.

[8] M. Chen, X. Tan, B. Li, Y. Liu, T. Qin, S. Zhao, and T.-Y. Liu, "AdaSpeech: Adaptive text to speech for custom voice," *arXiv preprint arXiv:2103.00993*, 2021.

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

[10] J. You, D. Kim, G. Nam, G. Hwang, and G. Chae, "Gan vocoder: Multi-resolution discriminator is all you need," *arXiv preprint arXiv:2103.05236*, 2021.

[11] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro, "DiffWave: A versatile diffusion model for audio synthesis," *arXiv preprint arXiv:2009.09761*, 2020.

[12] N. Chen, Y. Zhang, H. Zen, R. J. Weiss, M. Norouzi, and W. Chan, "WaveGrad: Estimating gradients for waveform generation," *arXiv preprint arXiv:2009.00713*, 2020.

[13] Y. Chung, Y. Wang, W. Hsu, Y. Zhang, and R. Skerry-Ryan, "Semi-supervised training for improving data efficiency in end-to-end speech synthesis," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2019, pp. 6940–6944.

[14] D. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *International Conference on Machine Learning*, 2015, pp. 1530–1538.

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

[16] D. P. Kingma and P. Dhariwal, "Glow: Generative flow with invertible 1x1 convolutions," *arXiv preprint arXiv:1807.03039*, 2018.

[17] B. Zhai, T. Gao, F. Xue, D. Rothchild, B. Wu, J. Gonzalez, and K. Keutzer, "SqueezeWave: Extremely lightweight vocoders for on-device speech synthesis," *arXiv preprint arXiv:2001.05685*, 2020.

[18] X. Tan, T. Qin, F. Soong, and T.-Y. Liu, "A survey on neural speech synthesis," *arXiv preprint arXiv:2106.15561*, 2020.

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

[20] H. Liao, J. He, and K. Shu, "Generative model with dynamic linear flow," *IEEE Access*, vol. 7, pp. 150 175–150 183, 2019.

[21] A. van den Oord, N. Kalchbrenner, L. Espeholt, K. Kavukcuoglu, O. Vinyals, and A. Graves, "Conditional image generation with pixelcnn decoders," in *Advances in Neural Information Processing Systems*, 2016, pp. 4797–4805.

[22] "Chinese standard mandarin speech corpus," https://www.databaker.com/data/index/source, Biao Bei (Beijing) Technology Co. Ltd.

[23] K. Ito and L. Johnson, "The LJ speech dataset," https://keithito.com/LJ-Speech-Dataset/, 2017.

[24] C. Lo, S. Fu, W. Huang, X. Wang, J. Yamagishi, Y. Tsao, and H. Wang, "MosNet: Deep learning based objective assessment for voice conversion," in *INTERSPEECH*, 2019, pp. 1541–1545.

[25] J. Wang, Y. You, F. Liu, D. Tuo, S. Kang, Z. Wu, and H. Meng, "The huya multi-speaker and multi-style speech synthesis system for m2voc challenge 2020," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2021, pp. 8608–8612.

[26] Y. Choi, Y. Jung, Y. Suh, and H. Kim, "Perceptually guided end-to-end text-to-speech with mos prediction," *arXiv preprint arXiv:2011.01174*, 2020.