FLOW-TTS: A NON-AUTOREGRESSIVE NETWORK FOR TEXT TO SPEECH BASED ON FLOW

Chenfeng Miao, Shuang Liang*, Minchuan Chen, Jun Ma, Shaojun Wang, Jing Xiao*

Ping An Technology

{miaochenfeng448, liangshuang161}@pingan.com.cn

ABSTRACT

In this work, we propose Flow-TTS, a non-autoregressive end-to-end neural TTS model based on generative flow. Unlike other non-autoregressive models, Flow-TTS can achieve high-quality speech generation by using a single feed-forward network. To our knowledge, Flow-TTS is the first TTS model utilizing flow in spectrogram generation network and the first non-autoregssive model which jointly learns the alignment and spectrogram generation through a single network. Experiments on LJSpeech show that the speech quality of Flow-TTS heavily approaches that of human and is even better than that of autoregressive model Tacotron 2 (outperforms Tacotron 2 with a gap of 0.09 in MOS). Meanwhile, the inference speed of Flow-TTS is about 23 times speed-up over Tacotron 2, which is comparable to FastSpeech. [1]

Index Terms— Text to speech, Non-autoregressive, Generative flow

1. INTRODUCTION

Text to speech (TTS) is an essential component of intuitive human-machine communication system. Given an input text sequence $\{x_1, x_2, ..., x_N\}$, TTS system can generate an output acoustic sequence $\{y_1, y_2, ..., y_T\}$. Concatenative TTS [1, 2] and statistical parametric TTS [3, 4] are the two most successful TTS techniques in the past decades. However, both of them have complex pipelines and the speech generated often sounds unnatural. As the rapid development of deep learning, lots of end-to-end TTS techniques have been proposed to achieve high-fidelity speech synthesis [5, 6, 7, 8, 9, 10]. These models usually consist of two parts. The first part is a spectrogram generation network, which transforms normalized text symbols to time-aligned features, such as mel-spectrogram. The second part is a vocoder, which transforms time-aligned features to audio samples. We focus on the first part in this work and use the WaveGlow [10] vocoder to generate audio samples. While the spectrogram generation networks vary widely from different aspects, they can be grouped into two categories, as shown in Table 1:

The first category is the autoregressive model (first col. in Table 1). These models can achieve a high speech quality but suffer from a low decoding speed. Although teacher forcing techniques [11] are often used in these models to speed up the training process and improve training stability, they cannot speed up the inference process and often lead to mismatches between different distributions of real and predicted data [12, 13].

The second category is the non-autoregressive model (second and third col. in Table 1). These models can speed up the inference process by means of parallel spectrogram generation. However, because of the difficulty of learning the alignment between text sequences and spectrogram sequences, these non-autoregressive models use a well-trained autoregressive teacher model to guide their training process, making the training process complex and unstable. Furthermore, non-autoregressive models often suffer from a low speech quality compared to autoregressive models. For example, FastSpeech [14] uses predicted mel-spectrograms from the teacher model as ground truth of the training. It stabilizes the training loss, while results in lower speech quality.

Autoregressive	Non-autoregressive	
	Need distillation	No distillation
Tacotron [5] Tacotron 2 [6] Deep Voice 3 [7] TransformerTTS [16]	ParaNet [15] FastSpeech [14]	**Flow-TTS**
WaveNet [9] WaveRNN [18] LPCNet [20]	ParallelWaveNet [17] ClariNet [8]	WaveGlow [10] FloWaveNet [19]

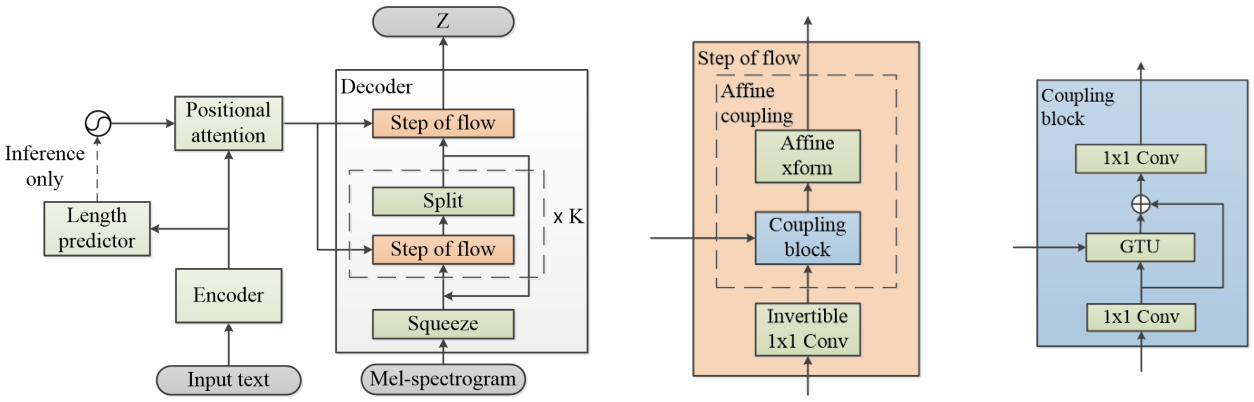
Table 1. End-to-end TTS models, including spectrogram generation models and vocoders.

In this work, we propose a novel approach towards non-autoregressive speech synthesis using a flow-based network. Our main contributions are as follows:

1. The first flow-based spectrogram generation network. Flow-TTS uses a powerful generative framework called generative flow (Glow) [21], which has obtained remarkable success in modeling images and speech through efficient density estimation and sampling [10].

*Equal contribution.

[1] Audio samples available at https://liangshuang1993.github.io/Flow-TTS-demo/index.html.



(a) Flow-TTS (b) Step of flow (c) Coupling block

Fig. 1. Overall model architecture. (a) Flow-TTS. (b) Step of flow. (c) Coupling block.

2. The first non-autoregssive model which jointly learns the alignment and spectrogram generation through a single feed-forward network, without using parameter distillation from teacher models.

3. State of the art speech quality. Series of experiments show that the speech quality of Flow-TTS is very close to human, and even better than that of existing autoregressive TTS models.

2. FLOW-BASED GENERATIVE MODEL

Flow-based generative model [21, 22] aims at transforming probability density from a simple probability density such as Gaussian distribution to a much more complex probability density by applying a sequence of invertible transformation functions. Formally, given a random variable $\mathbf{z}$ and its known probability density function $\mathbf{z} \sim \pi(\mathbf{z})$, flow-based generative models use a sequence of transformation functions $f = f_1 \circ f_2 \circ ... \circ f_L$ to map $\mathbf{z}$ to a new random variable $\mathbf{y}$ with the same dimension, where each f_i is invertible. The probability density function of $\mathbf{y}$ can be calculated using variable transformation:

$$\log\ p_Y(\mathbf{y}) = \log\ \pi(\mathbf{z}) + \sum_{i=1}^{L} \log \left| \det \frac{\partial f_i}{\partial f_{i-1}} \right|, \quad (1)$$

where $\det \frac{\partial f_i}{\partial f_{i-1}}$ is the Jacobian determinant of the function f_i. To make the calculation efficient, most flow-based generative models try to use triangular matrix as transformation Jacobian and use isotropic Gaussian as $\pi(\mathbf{z})$. In this way, flow-based generative models are capable of modeling high dimensional data by maximizing Eq.1 directly.

3. FLOW-TTS

Flow-TTS is based on generative flow (Glow) [21]. The overall architecture of Flow-TTS is shown in Fig. 1(a). It consists of an encoder, a decoder, a length predictor and a positional attention layer. We will describe these components in detail in the following subsections.

3.1. Encoder

Encoder first converts text symbols to trainable embeddings, and then followed by a series of convolutional blocks, each convolutional block consisting of a 1-D convolutional layer, a ReLU activation, a batch normalization and a dropout layer. An LSTM layer is added at the end of the encoder to extract long-range textual information. Given that the text length is much shorter than the output spectrogram length, LSTM layer would not affect the inference speed and parallel spectrogram generation, while greatly speed up the model convergence.

3.2. Length Predictor

Length predictor is used to predict the length of output spectrogram sequence. In autoregressive models, it is natural to predict the length of output spectrogram sequence by predicting a special stop token. However, Flow-TTS predicts all the output frames in parallel. Therefore, we have to predict the output length in advance. Length predictor consists of a 2-layer 1-D convolutional network, each followed by the layer normalization and dropout layer. An accumulated layer is added at the end of length predictor to accumulate all the symbol duration to the final length. We predict the length in logarithmic domain to get more stable training loss. Note that length predictor is stacked after the encoder and trained jointly with the rest of model.

3.3. Positional Attention

Positional attention layer is the key module to learn the alignment between input text sequences and output spectrogram sequences. It is implemented with a multi-head dot-product

attention mechanism [23], which takes the output hidden states of the encoder as the key vector and value vector, and takes the positional encoding of spectrogram length as query vector. Note that the spectrogram length is taken from the ground truth spectrogram during training, and predicted by the length predictor during inference.

3.4. Decoder

The decoder follows the architecture of Glow, which consists of a series of steps of flow along with the multi-scale architecture. Each step of flow consists of two invertible transformation layers, an invertible 1×1 convolutional layer and an affine coupling layer. We use a unique coupling block to feed the conditions generated from the positional attention layer into each step of spectrogram flow. We group the spectrogram along the channel dimension with a group size of 8, which serves as the "squeeze" operation described in [22].

Affine Coupling Layer. Affine coupling layer, first introduced in [21], is a powerful invertible transformation where both the forward function and reverse function are computationally efficient, as well as the log-determinant:

$$\mathbf{z_a}, \mathbf{z_b} = split(\mathbf{z}) \tag{2}$$

$$(\log \mathbf{s}, \mathbf{t}) = NN(\mathbf{z_b}) \tag{3}$$

$$\mathbf{s} = \exp(\log \mathbf{s}) \tag{4}$$

$$\mathbf{y_a} = \mathbf{s} \cdot \mathbf{z_a} + \mathbf{t} \tag{5}$$

$$\mathbf{y_b} = \mathbf{z_b} \tag{6}$$

$$\mathbf{y} = concat(\mathbf{y_a}, \mathbf{y_b}) \tag{7}$$

The $split()$ and $concat()$ function here are to split the input tensor into two halves and concatenate the output tensors together along the channel dimension. $NN()$ can be any nonlinear transformation, with no limitation of invertibility.

Invertible 1×1 Convolutional Layer. In order to permutate the ordering of the channels, a 1×1 invertible convolutional layer is plugged before the affine coupling layer. We initialize the weight matrix as a random orthonormal matrix, having a log-determinant of 0. The log-determinant would diverge from 0 after one training step.

Multi-Scale Architecture. Multi-scale architecture has been demonstrated helpful for training deep steps of flow. In our implementation, each scale is a combination of 4 steps of flow. After each scale, several channels of the tensor would be dropped from the flow steps and concatenated together after the model go through all the flow steps.

Coupling Block. Coupling block (in Fig. 1(c)) serves as $NN()$ transformation above. Coupling block starts with a 1-D convolutional layer with kernel size 1, and then followed by a modified gated tanh unit (GTU) [24] layer:

$$\mathbf{z} = \tanh(\mathbf{W_{f,k}} * \mathbf{y}) \odot \sigma(\mathbf{W_{g,k}} * \mathbf{c}), \tag{8}$$

where k denotes the layer index, f and g denote filter and gate respectively, W denotes the 1-D convolution, c denotes attention context vector. We add a residual connection over the GTU layer to fit for deep networks. We add another 1-D convolutional layer with kernel size 1 at the end to match channel dimensions. The weight of the last convolutional layer is initialized with zeros, which ensures each affine coupling layer initially performs an identity function. This kind of zero initialization greatly helps training deep flows.

4. EXPERIMENTS

We conduct experiments on LJSpeech dataset [25], which consists of 13100 English clips of a single speaker. Clips vary in duration from 1 to 10 seconds and have a total duration of approximately 24 hours. Each audio file is a mono channel 16-bit WAV with a sample rate of 22050 Hz, provided with a corresponding transcription. We randomly select 300 samples as the validation set and 300 samples as the test set. We follow the preprocess procedure in [6], which converts the raw waveforms to 80-channel mel-spectrograms. The parameters of the mel-spectrograms are FFT size 1024, window size 1024, hop size 256.

Our implementation of Flow-TTS is composed of 10 flow steps. Both the dimension size of character embeddings and the filter number of the convolutional layers in the encoder are set to 512. Positional attention layer has 4 heads with a hidden dimension size of 256. The drop probability of dropout layer is set to 0.2 in the encoder, 0.3 in the length predictor, 0.1 in the coupling block and the positional attention layer. We drop 2 groups for every 4 flow steps when implementing the multi-scale architecture. WaveGlow is utilized as the vocoder to produce final waveforms. To evaluate the performance of our model, we choose a typical autoregressive model Tacotron 2 and a typical non-autoregressive model FastSpeech as baseline models.

MOS Evaluation. We randomly select 20 audio samples from the test set and conduct a mean opinion score (MOS) evaluation to evaluate audio naturalness. Audio generated by 4 other models are mixed with the original audio. There are 12 testers who are all well-educated in English. The results are shown in Table 2 with 95% confidence intervals. We can see that Flow-TTS outperforms other TTS models even including the autoregressive model.

Model	MOS
Ground Truth	4.55 ± 0.27
Ground Truth (Mel + WaveGlow) [10]	4.35 ± 0.28
Tacotron 2 [6] (Mel + WaveGlow)	4.10 ± 0.37
FastSpeech [14] (Mel + WaveGlow)	3.79 ± 0.32
Flow-TTS (Mel + WaveGlow)	$\mathbf{4.19 \pm 0.28}$

Table 2. MOS results of different models. Higher is better.

MCD and F0 Trajectories. We notice the prosody of the generated speech by Flow-TTS is very close to that of ground truth. See Fig. 2. To address this further, we conduct an ob-

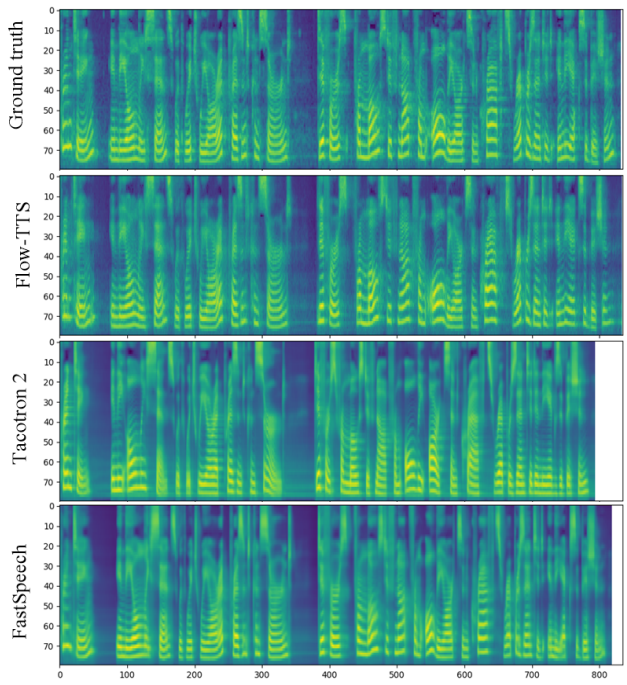


Fig. 2. Mel-spectrogram of Flow-TTS is more closer to GT than other TTS models. Best viewed in color.

Model	MCD
Tacotron 2	6.73
FastSpeech	5.81
Flow-TTS	**5.43**

Table 3. MCD results of different models. Lower is better.

jective evaluation using mel cepstral distortions (MCD) [26]. We randomly select 50 sentences from the test dataset, the results are shown in Table 3. Flow-TTS achieves the best MCD score[2]. We also illustrate F0 trajectories for different models in Fig. 3. Both of the above experiments showing that the speech generated by Flow-TTS is more closer to that of ground truth than other TTS models.

Inference Speed. We compare the inference latency of Flow-TTS with autoregressive Tacotron 2. We conduct a 20 sentence test set and run the inference for 20 times on NVIDIA Tesla V100. The length of the generated mel-spectrograms is range from 437 to 937, with an average of 680. The average inference latency is 0.021 and 0.483 seconds for Flow-TTS and Tacotron 2. Therefore, Flow-TTS brings about 23 times speed-up compared to Tacotron 2. Meanwhile, our experiment results indicate that Flow-TTS is faster than FastSpeech, which has a inference latency of 0.025 seconds with an average spectrogram length of 560, as described in their paper.

[2]We notice that FastSpeech gets better MCD score than autoregressive Tacotron 2, it might be because FastSpeech predicts the alignment in parallel, similar as Flow-TTS, although it primarily learns the alignment from a teacher autoregressive model.

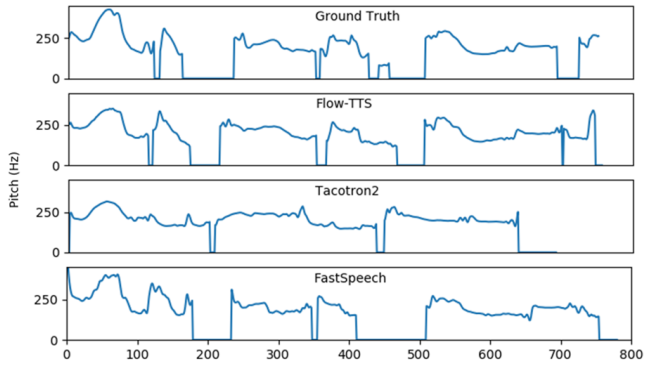


Fig. 3. F0 trajectories of the utterance "The Roman type of all these printers is similar in character,"

5. DISCUSSION

It is challenging to learn the alignment between text sequences and spectrogram sequences for TTS models. Existing non-autoregressive models use knowledge distillation approaches to extract the information from autoregressive models, making the training process complex and unstable. Autoregressive models use teacher forcing techniques to guide their training process, which greatly speed up the training process, while still suffering from a low inference speed and a declining quality of synthesis caused by mismatch between training and inference.

In this work, we introduce Flow-TTS, a flow-based TTS model. By constraining the network to be invertible, flow-based models give us a tractable way to model the distribution of data. Specifically, Flow-TTS is based on Glow. Although we notice that there are some other generative flow frameworks proposed recently, such as Flow++ [27], we choose Glow for the following reasons. Firstly, Glow has obtained great success in modeling audio generation [10]. Secondly, the invertible 1×1 convolutional layer and the affine coupling layer with zero weight initialization of Glow greatly help train relatively deep flows more stably. Finally, the self-attention [23] based coupling layer in Flow++ would lead to more attention errors and more computation cost, compared with our CNN based coupling layer.

6. CONCLUSION

Flow-TTS is a fast and high quality TTS model, which allows for efficient parallel spectrogram generation along with jointly learning the alignment between input text sequences and output spectrogram sequences. For future work, we would apply Flow-TTS to Voice Conversion and Voice Cloning as flow-based models have shown great ability to preserve the characteristics of the original audio data.

7. REFERENCES

[1] A. J. Hunt and A. W. Black, "Unit selection in a concatenative speech synthesis system using a large speech database," in *ICASSP*, 1996, pp. 373–376.

[2] A. W. Black and P. Taylor, "Automatically clustering similar units for unit selection in speech synthesis," in *Eurospeech*, 1997, pp. 601–604.

[3] K. Tokuda, T. Yoshimura, T. Masuko, T. Kobayashi, and T. Kitamura, "Speech parameter generation algorithms for hmm-based speech synthesis," in *ICASSP*, 2000, vol. 3, pp. 1315–1318.

[4] H. Ze, A. Senior, and M. Schuster, "Statistical parametric speech synthesis using deep neural networks," in *ICASSP*, 2013, pp. 7962–7966.

[5] Y. Wang, R. Skerry-Ryan, D. Stanton, R. J. Weiss Y. Wu, N. Jaitly, and Z. Yang, "Tacotron: Towards end-to-end speech synthesis," in *Interspeech*, 2017.

[6] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, and Y. Zhang Y. Wang R. SkerryRyan Z. Yang, Z. Chen, "Natural TTS synthesis by conditioning Wavenet on mel spectrogram predictions," in *ICASSP*, 2018, pp. 4779–4783.

[7] W. Ping, K. Peng, A. Gibiansky, S. O. Arik, A. Kannan, S. Narang, J. Raiman, and J. Miller, "Deep Voice 3: 2000-speaker neural text-to-speech," in *ICLR*, 2018.

[8] W. Ping, K. Peng, and J. Chen, "Clarinet: Parallel wave generation in end-to-end text-to-speech," in *ICLR*, 2019.

[9] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," in *9th ISCA Speech Synthesis Workshop*, 2016, pp. 125–125.

[10] R. Prenger, R. Valle, and B. Catanzaro, "Waveglow: A flow-based generative network for speech synthesis," in *ICASSP*, 2019, pp. 3617–3621.

[11] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural computation*, vol. 1, no. 2, pp. 270–280, 1989.

[12] A. M. Lamb, A. G. A. P. GOYAL, Y. Zhang, S. Zhang, A. C. Courville, and Y. Bengio, "Professor forcing: A new algorithm for training recurrent networks," in *NeurIPS*, 2016, pp. 4601–4609.

[13] H. Guo, F. K. Soong, L. He, and L. Xie, "A new gan-based end-to-end TTS training algorithm," in *Interspeech*, 2019, pp. 1288–1292.

[14] Y. Ren, Y. Ruan, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T.-Y. Liu, "Fastspeech: Fast, robust and controllable text to speech," in *NeurIPS*, 2019.

[15] K. Peng, W. Ping, Z. Song, and K. Zhao, "Parallel neural text-to-speech," 2019, vol. abs/1905.08459.

[16] N. Li, S. Liu, Y. Liu, S. Zhao, M. Liu, and M. Zhou, "Close to human quality TTS with transformer," in *AAAI*, 2019.

[17] A. van den Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. v. d. Driessche, E. Lockhart, et al., "Parallel WaveNet: Fast high-fidelity speech synthesis," in *PMLR*, 2018.

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

[19] S. Kim, S. Lee, J. Song, J. Kim, and S. Yoon, "FloWaveNet: A generative flow for raw audio," in *ICML*, 2019.

[20] J. Valin and J. Skoglund, "LPCNet: Improving neural speech synthesis through linear prediction," in *ICASSP*, 2019.

[21] D. P. Kingma and P Dhariwal, "Glow: Generative flow with invertible 1x1 convolutions," in *NeurIPS*, 2018, pp. 10236–10245.

[22] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real NVP," in *ICLR*, 2017.

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, Lukasz Gomez, A. N. Gomez, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, 2017, pp. 5998–6008.

[24] A. van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, et al., "Conditional image generation with pixelcnn decoders," in *NeurIPS*. 2016.

[25] K. Ito, "The LJ speech dataset," `https://keithito.com/LJ-Speech-Dataset/`, 2017.

[26] R. Kubichek, "Mel-cepstral distance measure for objective speech quality assessment," in *IEEE Pacific Rim Conference on Communications Computers and Signal Processing*, 1993, vol. 1, pp. 125–128.

[27] J. Ho, X. Chen, A. Srinivas, Y. Duan, and P. Abbeel, "Flow++: Improving flow-based generative models with variational dequantization and architecture design," in *ICML*, 2019.