VARIANCEFLOW: HIGH-QUALITY AND CONTROLLABLE TEXT-TO-SPEECH USING VARIANCE INFORMATION VIA NORMALIZING FLOW

Yoonhyung Lee[1*] *Jinhyeok Yang*[2] *Kyomin Jung*[1]

[1]Seoul National University, Dept. of Electrical and Computer Engineering, Republic of Korea
[2]Speech AI Lab, NCSOFT, Republic of Korea

ABSTRACT

There are two types of methods for non-autoregressive text-to-speech models to learn the one-to-many relationship between text and speech effectively. The first one is to use an advanced generative framework such as normalizing flow (NF). The second one is to use variance information such as pitch or energy together when generating speech. For the second type, it is also possible to control the variance factors by adjusting the variance values provided to a model. In this paper, we propose a novel model called VarianceFlow combining the advantages of the two types. By modeling the variance with NF, VarianceFlow predicts the variance information more precisely with improved speech quality. Also, the objective function of NF makes the model use the variance information and the text in a disentangled manner resulting in more precise variance control. In experiments, VarianceFlow shows superior performance over other state-of-the-art TTS models both in terms of speech quality and controllability.

***Index Terms*—** Text-to-Speech, Speech Synthesis, Normalizing Flow, Generative Model

1. INTRODUCTION

In Text-to-Speech (TTS), the one-to-many relationship between text and speech is one of the major problems that makes it challenging to learn the text-to-speech conversion. The early autoregressive (AR) TTS models [1, 2] dealt with the difficulty by factorizing the speech distribution into the product of homogeneous conditional factors in sequential order. However, although they succeeded in generating high-quality speech, their slow inference speed and exposure bias were inevitable problems inherent in the AR models.

As it has advanced from AR TTS models to non-AR TTS models, two types of methods for the non-AR models to solve the one-to-many problem have been proposed. The first type (Type-I) is to use an advanced generative framework such as normalizing flow [3], diffusion model [4], and generative adversarial network [5]. Unlike the mean squared error (MSE) based training that assumes a Gaussian distribution, these frameworks do not assume any pre-defined distribution

*This work is done during an internship program at NCSOFT.

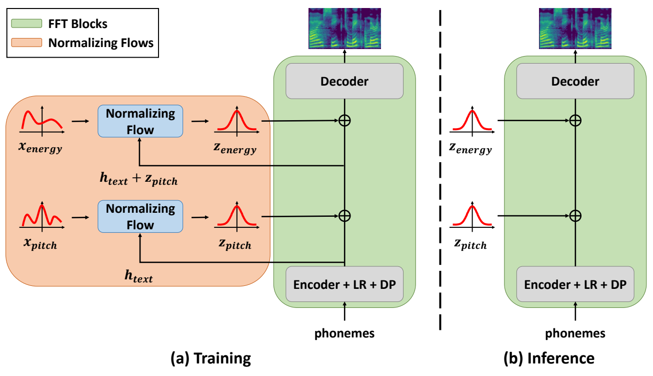

Fig. 1. The architecture of VarianceFlow. LR denotes a length regulator that expands text representations to the length of a melspectrogram based on phoneme durations. DP denotes a duration predictor that predicts the phoneme durations.

for a target distribution. As a result, they can generate high-quality and diverse speech samples compared to the previous non-AR TTS models trained with the MSE loss [6, 7]. The second type of the methods (Type-II) is to solve a task of TTS by dividing it into simpler two tasks based on variance information such as pitch or energy [8, 9, 10]: (1) text conditioned variance modeling; (2) text and variance information conditioned speech generation. Then, by using ground-truth variance information when learning the two tasks, Type-II models achieve faster training convergence and higher speech quality. Moreover, unlike Type-I models, Type-II models can explicitly control the variance factors by manipulating the variance values used in the speech generation.

In this paper, we propose VarianceFlow, a novel model that combines the advantages of Type-I and Type-II, achieving a high-quality and controllable TTS model. Unlike the previous Type-II models that learn the text conditioned variance modeling based on the MSE loss, our model uses normalizing flow (NF) for the variance modeling (Figure 1).

There are two large advantages of using NF for the variance modeling. First, since NF is robust to the one-to-many problem, it learns the variance distribution better than using the MSE loss [11], and it leads to improved speech quality. Second, NF enhances the variance controllability by disentangling the latent variance representation and the text [12].

In experiments, VarianceFlow shows its superiority in variance modeling by showing better speech quality compared to other AR and non-AR state-of-the-art TTS models. In addition, VarianceFlow shows its more precise variance controllability by showing that it uses the provided pitch values more intactly. Lastly, we show that VarianceFlow can generate diverse speech samples given a text input.

2. VARIANCEFLOW

In this section, we first explain FastSpeech 2 [8], which is a baseline of our model. Then, we present VarianceFlow that learns to match a variance distribution to a prior distribution based on normalizing flow.

2.1. FastSpeech 2

FastSpeech 2 is a Type-II non-AR TTS model that uses pitch and energy as additional variance information in speech generation[1]. Firstly, a phoneme sequence is encoded by an FFT encoder, and it is expanded to the length of the target melspectrogram given phoneme durations. Next, ground-truth pitch and energy values extracted in frame-level are projected to the space of the text representation. Lastly, the projected variance vectors are added to the expanded text representation, and an FFT decoder generates a melspectrogram based on it.

Since the ground-truth variance information is unavailable at inference, FastSpeech 2 has additional modules called variance predictor for variance modeling. During training, the variance predictors are trained with the MSE loss to predict the ground-truth variance values based on the input text. Then, the variance values predicted by the variance predictors are used at inference. However, there is still a problem that predicting pitch or energy from a text is also a difficult task having a one-to-many relationship. For this, FastSpeech 2 trains its pitch predictor with wavelet transformed pitch spectrogram instead of raw pitch values. Also, FastPitch [9], which uses pitch information similar to FastSpeech 2, uses phoneme-averaged pitch values to ease the difficulty of learning the complex pitch distribution.

2.2. VarianceFlow

VarianceFlow is a model that uses variance information such as pitch and energy based on normalizing flow [13, 11]. During training, pitch and energy variance information is provided to a feed-forward transformer (FFT) decoder via NF modules (Figure 1). Then, VarianceFlow learns to generate speech based on the information and the input text. Meanwhile, the NF modules learn to match a latent variance distribution to a simple prior distribution. At inference, the variance information is provided to the FFT blocks by directly sampling the latent representation from the prior.

A normalizing flow (NF) consisting of consecutive bijective transforms converts a complex variance distribution to a simple prior distribution. Based on the change of variables formula, using bijective transforms enables to calculate probability density of the variance information by setting the latent variance distribution as a simple prior distribution:

$$\log \boldsymbol{p_\theta}(\boldsymbol{x}|\boldsymbol{h}) = \log \boldsymbol{p}(\boldsymbol{z}) + \sum_{i=1}^{k} \log|\det(\boldsymbol{J}(\boldsymbol{f_i}(\boldsymbol{x};\boldsymbol{h})))| \quad (1)$$

$$\boldsymbol{z} = \boldsymbol{f_k} \circ \boldsymbol{f_{k-1}} \circ \ldots \circ \boldsymbol{f}_0(\boldsymbol{x};\boldsymbol{h}), \quad (2)$$

where $\boldsymbol{x}$ is a variance factor, $\boldsymbol{h}$ is a hidden representation, $\mathbf{z}$ is a latent representation, and $\boldsymbol{f_i}$ is a bijective transform. Therefore, using a unit Gaussian distribution as the prior distribution, VarianceFlow is trained to maximize the log-likelihood of the variance information based on (1) and (2). To fully utilize parallel computation with enough expressive power, VarianceFlow uses rational-quadratic coupling transform [14].

2.3. Loss function

Replacing the MSE losses for pitch and energy predictors in FastSpeech 2 [8] with the NF losses of VarianceFlow, the final objective function of VarianceFlow becomes as follows:

$$\mathcal{L}_{total} = \mathcal{L}_{melspec} + \mathcal{L}_{duration} + \alpha \cdot \mathcal{L}_{pitch} + \alpha \cdot \mathcal{L}_{energy}, \quad (3)$$

where the first two terms are the losses existing in FastSpeech 2, and the last two terms are the negative log-likelihood NF losses for pitch and energy. Here, each of the NF losses ($\mathcal{L}_{NF}$) not only learns the variance modeling but also disentangles the variance factor with the text, and it is shown by the fact that $\mathcal{L}_{NF}$ can be decomposed as follows [12]:

$$\mathcal{L}_{NF} = D_{KL}\left[\boldsymbol{q_\theta}(\boldsymbol{z}|\boldsymbol{h}) \parallel \boldsymbol{p}(\boldsymbol{z})\right] + H(\boldsymbol{x}|\boldsymbol{h}) \quad (4)$$

For an NF module in VarianceFlow, the second entropy term is constant, so it is trained to minimize the kullback-leibler divergence between the conditional latent variance distribution $\boldsymbol{q_\theta}(\boldsymbol{z}|\boldsymbol{h})$ and the prior distribution $\boldsymbol{p}(\boldsymbol{z})$. It means that it is trained to make $\boldsymbol{z}$ and $\boldsymbol{h}$ disentangled because the prior $\boldsymbol{p}(\boldsymbol{z})$ is chosen independently with $\boldsymbol{h}$. As a result, it enhances the responsiveness of the model to $\boldsymbol{z}$, resulting in more precise control.

2.4. Controlling a variance factor

The invertibility of NF makes it possible to obtain the raw variance value $\boldsymbol{x}$ from the latent representation $\boldsymbol{z}$. Therefore, it is also possible for VarianceFlow to control the variance factors as other Type-II TTS models (Figure 2). To control the variance information, the sampled latent representations are first brought to the raw variance space using the inverse

[1]Phoneme durations are also used as variance information, but it is out of focus of this work.

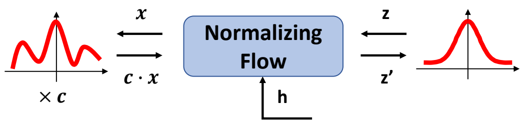

Fig. 2. Flow of signals when controlling a variance factor

transforms of NF. Then, the variance factor is manipulated in the space (*e.g.*, multiplying a constant to the raw variance value $\boldsymbol{x}$). Lastly, the manipulated value is provided to the FFT decoder via NF again.

3. EXPERIMENTS

In this section, experimental setup and results are provided. Also, the audio samples and more experimental figures are available at `https://leeyoonhyung.github.io/VarianceFlow-demo`.

3.1. Experimental setup

In experiments, LJSpeech dataset [15] is used. The waves are converted to log-melspectrograms with 1024 fft window and 256 hop lengths. The pitch values are extracted using Parselmouth [16] and they are converted to log-scale and pitch spectrogram is not used. When calculating the total loss of VarianceFlow, $\alpha = 0.1$ is used for the coefficient of the NF losses. When generating audio samples, we use a zero-mean Gaussian prior distribution with a standard deviation $\sigma = 0.333$, and we use HiFi-GAN vocoder [17]. As a baseline, we adopt FastSpecch 2 architecture [8] and also adopt its FFT architecture for VarianceFlow. For an NF module of VarianceFlow, we use 4-layer rational-quadratic coupling transform architecture following the architecture of the stochastic duration predictor in [18]. We train all models for 230k steps with a batch size of 16. We use AdamW optimizer [19] with $\beta_1 = 0.9$ and $\beta_2 = 0.98$ with the Noam learning rate scheduling [20].

3.2. Speech quality

To see the effectiveness of VarianceFlow in improving speech quality, we compare the speech quality of VaraineceFlow with FastSpeech 2 based on the 9-scale mean opinion score (MOS), while varying the ways of using the variance information either phoneme-averaged level or frame level (Table 1). For each model, fifty samples from the test set are used, and five testers living in the U.S. are asked to give a score ranging from 1 to 5 to each audio sample. For FastSpeech 2, MOS is better when variance information is used in the phoneme-averaged level. It means that training the current architecture of the variance predictors using the MSE loss lacks in learning the complex variance distribution. On the contrary, VarianceFlow shows better speech quality when the variance information is provided in the frame level. It means

Table 1. 9-scale MOS results with 95% confidence intervals.

Model	MOS
GT Waveform	4.47 ± 0.07
GT Melspectrogram	4.34 ± 0.08
Tacotron 2	4.03 ± 0.07
Glow-TTS	3.72 ± 0.13
FastSpeech 2-phoneme	3.92 ± 0.07
FastSpeech 2-frame	3.66 ± 0.09
VarianceFlow-phoneme	4.04 ± 0.08
VarianceFlow-frame	$\mathbf{4.19 \pm 0.07}$

that our VarianceFlow learns the complex variance distribution well based on NF. Although the architectures used in variance modeling are different, directly using the latent representation sampled from a Gaussian distribution does not slow down the synthesis speed or increase the number of parameters at inference. Overall, VarianceFlow trained with the frame level variance information shows the best speech quality including other state-of-the-art AR and non-AR TTS models, Tacotron 2 and Glow-TTS.

3.3. Controllability

To compare the controllability of the models, we conduct objective and subjective evaluations while adjusting the predicted pitch values with various pitch shift coefficients λ. In addition, to see the effectiveness of the NF loss in disentangling the variance factors with other factors, we also train a model by reverting the direction of NF ('VarianceFlow-reversed'). For this model, the raw variance information is directly provided to the FFT decoder while the NF modules separately learn to match the latent variance distribution and the prior distribution. In this case, it still has the advantage of using advanced distribution modeling method, but it uses the variance information without disentangling.

3.3.1. Pitch responsiveness

To evaluate the responsiveness of the models to the variance information, we measure f0 frame error rate (FFE) [21] between the pitch values provided to the decoder and the pitch values extracted from the generated audio samples. All audio samples are generated by adjusting the pitch values in a semitone unit as follows [10]:

$$f_\lambda = 2^{\frac{\lambda}{12}} \times f_0$$

where f_0 is the pitch value before shifted. In the experiments, $\lambda = \{-6, -4, -2, +2, +4, +6\}$ are used. As shown in Table 2 and Figure 3, VarianceFlow uses the provided pitch most intactly for most λ. Therefore, it is possible to control the pitch with VarianceFlow precisely. VarianceFlow-reversed shows

Table 2. FFE (%) and 9-scale MOS results measured with different pitch shift scale λ.

Model	$\lambda = -6$		$\lambda = -4$		$\lambda = -2$		$\lambda = +2$		$\lambda = +4$		$\lambda = +6$	
	FFE	MOS	FFE	MOS	FFE	MOS	FFE	MOS	FFE	MOS	FFE	MOS
FastSpeech 2	15.94	2.93	14.00	3.46	12.61	3.65	10.94	3.29	11.57	2.63	12.24	2.04
VarianceFlow-reversed	21.93	3.77	35.97	4.01	53.47	4.00	66.37	3.90	67.07	3.69	67.03	3.53
VarianceFlow	24.14	3.43	12.16	3.87	9.02	4.05	7.26	3.95	7.52	3.39	7.68	2.74

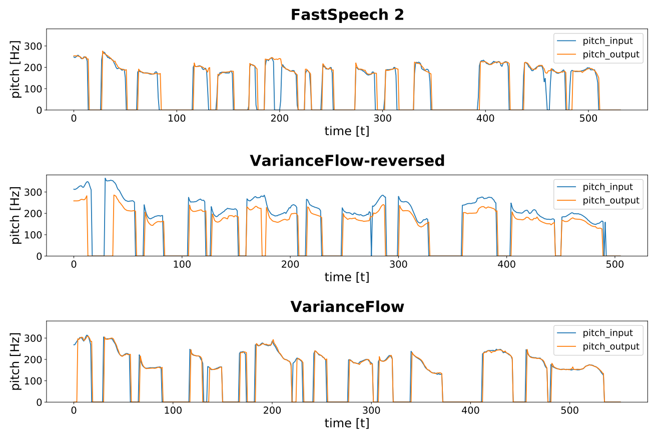

Fig. 3. Each figure shows f0 contours of input pitch and extracted pitch when $\lambda = 0$.

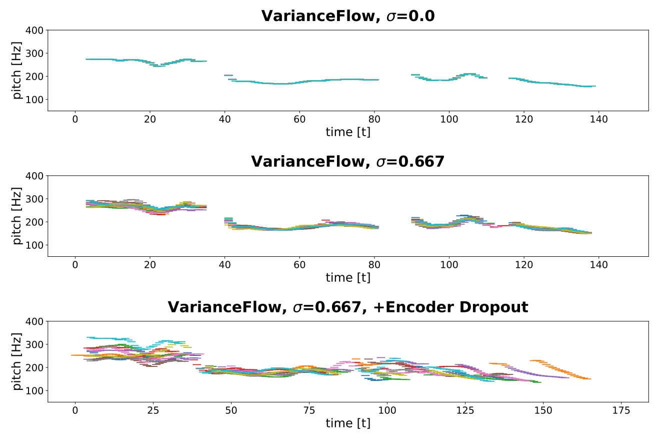

Fig. 4. Each figure shows f0 contours of 10 different samples generated using the same σ.

the largest FFE, and it can be understood based on Figure 3. Figure 3 shows that VarianceFlow-reversed learns to use the variation of provided pitch, but it ignores the average of the provided pitch when generating a speech.

3.3.2. *Speech quality with pitch shift*

To evaluate the models subjectively, we measure the MOS for the samples generated using the shifted pitch values in the same way with Section 3.2. For each model and λ, twenty samples from the test set are used. The results in Table 2 show that our VarianceFlow achieves better speech quality for all λ compared to FastSpeech 2, and even outperforms VarianceFlow-reversed for $\lambda = \{-2, +2\}$. We conjecture that this is because while the latent representation is passed to the FFT decoder, NF loss works as a regularization. Although VarianceFlow becomes worse than VarianceFlow-reversed when λ gets larger, we attribute this to the fact that VarianceFlow-reversed does not follow the pitch shift.

3.4. Diversity

To see how the variation in latent space affects the variance factors, we generate diverse speech samples from a single text while using different standard deviation $\sigma = \{0.0, 0.667\}$ for latent sampling (Figure 4). Here, the σ values are chosen in a range where the speech quality is maintained. In the experiment, it is shown that VarianceFlow can generate natural speech without using latent representations, and pitch variance in samples appears when using the latent representations. However, the different samples have similar prosody and perceptually sound similar. We conjecture that this is because it is trained to minimize the conditional entropy $H(\boldsymbol{x}|\boldsymbol{h})$ (Eq. (4)). Therefore, we repeat the samplings while having the dropout layers in the FFT encoder activated [18, 22]. In this case, the diversity in terms of prosody increases with varying durations without much loss of speech quality.

4. CONCLUSION

We propose VarianceFlow, a novel model that uses variance information based on normalizing flow (NF). VarianceFlow shows that it learns the variance distribution better with improved speech quality. Also, the invertibility of NF maintains its ability to control the variance factors, and we show that the objective of NF even enhances the variance controllability.

5. ACKNOWLEDGMENTS

K. Jung is with ASRI, Seoul National University, Korea. This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (No. 2021R1A2C2008855). This research resulted from a study on the "HPC Support" Project, supported by the 'Ministry of Science and ICT' and NIPA.

6. REFERENCES

[1] Jonathan Shen, Ruoming Pang, Ron J Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, et al., "Natural tts synthesis by conditioning wavenet on mel spectrogram predictions," in *Proc. ICASSP*, 2018, pp. 4779–4783.

[2] Naihan Li, Shujie Liu, Yanqing Liu, Sheng Zhao, and Ming Liu, "Neural speech synthesis with transformer network," in *Proc. of the AAAI conference on Artificial Intelligence*, 2019, vol. 33, pp. 6706–6713.

[3] Jaehyeon Kim, Sungwon Kim, Jungil Kong, and Sungroh Yoon, "Glow-tts: A generative flow for text-to-speech via monotonic alignment search," in *Proc. Advances in Neural Information Processing Systems*, 2020, vol. 33, pp. 8067–8077.

[4] Vadim Popov, Ivan Vovk, Vladimir Gogoryan, Tasnima Sadekova, and Mikhail Kudinov, "Grad-tts: A diffusion probabilistic model for text-to-speech," in *Proc. Int. Conf. on Machine Learning (ICML)*, 2021, vol. 139, pp. 8599–8608.

[5] Jinhyeok Yang, Jae-Sung Bae, Taejun Bak, Young-Ik Kim, and Hoon-Young Cho, "GANSpeech: Adversarial Training for High-Fidelity Multi-Speaker Speech Synthesis," in *Proc. Interspeech*, 2021, pp. 2202–2206.

[6] Yi Ren, Yangjun Ruan, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, et al., "Fastspeech: Fast, robust and controllable text to speech," in *Proc. Advances in Neural Information Processing Systems*, 2019, vol. 32.

[7] Kainan Peng, Wei Ping, Zhao Song, and Kexin Zhao, "Non-autoregressive neural text-to-speech," in *Proc. Int. Conf. on Machine Learning (ICML)*, 2020, pp. 10192–10204.

[8] Yi Ren, Chenxu Hu, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, et al., "Fastspeech 2: Fast and high-quality end-to-end text to speech," in *Proc. Int. Conf. on Learning Representations (ICLR)*. 2021, OpenReview.net.

[9] Adrian Łańcucki, "Fastpitch: Parallel text-to-speech with pitch prediction," in *Proc. ICASSP*, 2021, pp. 6588–6592.

[10] Taejun Bak, Jae-Sung Bae, Hanbin Bae, Young-Ik Kim, and Hoon-Young Cho, "FastPitchFormant: Source-Filter Based Decomposed Modeling for Speech Synthesis," in *Proc. Interspeech*, 2021, pp. 116–120.

[11] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio, "Density estimation using real NVP," in *Proc. Int. Conf. on Learning Representations (ICLR)*. 2017, OpenReview.net.

[12] Robin Rombach, Patrick Esser, and Bjorn Ommer, "Network-to-network translation with conditional invertible neural networks," in *Proc. Advances in Neural Information Processing Systems*, 2020, vol. 33, pp. 2784–2797.

[13] Laurent Dinh, David Krueger, and Yoshua Bengio, "NICE: non-linear independent components estimation," in *3rd Int. Conf. on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*, Yoshua Bengio and Yann LeCun, Eds., 2015.

[14] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios, "Neural spline flows," in *Proc. Advances in Neural Information Processing Systems*, 2019, vol. 32.

[15] Keith Ito and Linda Johnson, "The lj speech dataset," `https://keithito.com/LJ-Speech-Dataset/`, 2017.

[16] Yannick Jadoul, Bill Thompson, and Bart de Boer, "Introducing Parselmouth: A Python interface to Praat," *Journal of Phonetics*, vol. 71, pp. 1–15, 2018.

[17] Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae, "Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis," in *Proc. Advances in Neural Information Processing Systems*, 2020, vol. 33, pp. 17022–17033.

[18] Jaehyeon Kim, Jungil Kong, and Juhee Son, "Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech," in *Int. Conf. on Machine Learning (ICML)*, 2021, vol. 139 of *Proceedings of Machine Learning Research*, pp. 5530–5540.

[19] Ilya Loshchilov and Frank Hutter, "Decoupled weight decay regularization," in *Proc. Int. Conf. on Learning Representations (ICLR)*. 2019, OpenReview.net.

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, et al., "Attention is all you need," in *Proc. Advances in Neural Information Processing Systems*, 2017, vol. 30.

[21] Wei Chu and Abeer Alwan, "Reducing f0 frame error of f0 tracking algorithms under noisy conditions with an unvoiced/voiced classification frontend," in *2009 IEEE Int. Conf. on Acoustics, Speech and Signal Processing*. IEEE, 2009, pp. 3969–3972.

[22] Rafael Valle, Kevin J. Shih, Ryan Prenger, and Bryan Catanzaro, "Flowtron: an autoregressive flow-based generative network for text-to-speech synthesis," in *Proc. Int. Conf. on Learning Representations (ICLR)*, 2021.