KimchiSpeech: Faster, Lighter and More Controllable TTS with Hierarchical VAE and STT-aided Gaussian Attention

Soochul Park

MODULABS, Seoul, Korea

scpark20@gmail.com

Abstract

In this study, we propose a text-to-speech (TTS) model, referred to as KimchiSpeech, which has a hierarchical variational autoencoder (VAE) structure and uses a attention alignment obtained from a speech-to-text (STT) model. The hierarchical VAE structure contributes to the generation of high-quality outputs and a variety of prosody. Because the STT model operates independently of the TTS model, the attention alignment can be obtained robustly. Moreover, the speed of a generated speech can be flexibly controlled by soft attention using Gaussian distributions. Furthermore, we propose two configurations of the model, namely KimchiSpeech-W and KimchiSpeech-S. The former is a light version that has only 3.3M parameters for inference, whereas the latter is a fast version that produces outputs 470 times faster than real time on a GPU. The mean opinion score (MOS) results show that the outputs are of state-of-the-art quality. [1]

Index Terms: text-to-speech, speech synthesis, variational autoencoder, attention

1. Introduction

After the advent of Tacotron [1, 2], end-to-end text-to-speech (TTS) using neural networks are actively studied. Since Tacotron generates mel-spectrogram sequentially as an autoregressive model, the inference is slow. Several non-autoregressive models, such as FastSpeech [3], have been proposed to address the problem of slow inference. Non-autoregressive models can be partitioned into regression-based, flow-based, and generative adversarial network (GAN) [4]-based models.

Regression-based models include FastSpeech and its successor, FastSpeech2 [5]. Although TTS is a one-to-many function that does not fit with the regression method, but it can be addressed using Transformer [6] with a powerful inference capability. In FastSpeech2, the quality is improved by using additional information such as pitch and energy of voice.

Flow-based models using normalizing flows include Flow-TTS [7] and Glow-TTS [8]. They transform a given data distribution into a prior distribution using a sequence of bijective functions. Although they have a disadvantage that all layers in the networks should be invertible, they enable the production of outputs with various prosody by sampling in the prior distribution.

Variational autoencoder (VAE)-based models include BVAE-TTS [9] and VARA-TTS [10]. The flow-based models must have invertible layers, whereas the VAE-based models address this limitation by introducing the variational posterior. Similar to flow-based models, they can sample from the prior distribution.

Additionally, there are GAN-TTS [11], EATS [12] and FastSpeech2s [5] that directly generate waveforms using GAN.

KimchiSpeech is a VAE-based model that follows the architecture of very deep VAE (VDVAE) [13], a type of hierarchical VAE (Section 2.2). By simplifying the structure, the inference speed can be enhanced.

Our contributions are summarized as follows:

- We applied Hierarchical VAE to speech synthesis, enabling a rapid generation of high-quality outputs.
- We propose a speech-to-text (STT)-aided Gaussian attention that operates independently of the TTS to generate a robust attention alignment, and allows flexible control of the length of the speech.
- We provide a weight optimized model and a speed optimized model to enable usage in a several applications.

2. Background

2.1. Variational Inference

VAE [14], a type of generative model, consists of an encoder and a decoder. These are used to represent the variational posterior $q_\phi(\mathbf{z}|\mathbf{x})$ and the conditional distribution $p_\theta(\mathbf{x}|\mathbf{z})$, respectively. The objective function is the lower bound of the log-likelihood of the data $\mathbf{x}$.

$$\log p_\theta(\mathbf{x}) \geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z})) \tag{1}$$

The right side of equation 1 is called the evidence lower bound (ELBO) and consists of the reconstruction term and the Kullback–Leibler (KL) divergence term. The reconstruction term enables the reconstruction of the data source $\mathbf{x}$ given $\mathbf{z}$ from the variational posterior $q_\phi(\mathbf{z}|\mathbf{x})$. The KL-divergence term makes the variational posterior $q_\phi(\mathbf{z}|\mathbf{x})$ close to the prior $p_\theta(\mathbf{z})$.

2.2. Hierarchical VAE

In the VAE, the prior is assumed to be a fully factorized Gaussian distribution. This limitation is unreasonable for encoding the data distribution, resulting in poor quality outputs. To address this, models with hierarchical structures such as Ladder-VAE [15], NVAE [16], VDVAE [13] were proposed. These models factorize the prior $p_\theta(\mathbf{z})$ hierarchically by the chain rule.

$$p_\theta(\mathbf{z}) = p_\theta(\mathbf{z}_0)p_\theta(\mathbf{z}_1|\mathbf{z}_0)...p_\theta(\mathbf{z}_N|\mathbf{z}_{<N}) \tag{2}$$

The variational posterior $q_\phi(\mathbf{z}|\mathbf{x})$ is also factorized as follows.

$$q_\phi(\mathbf{z}|\mathbf{x}) = q_\phi(\mathbf{z}_0|\mathbf{x})q_\phi(\mathbf{z}_1|\mathbf{z}_0,\mathbf{x})...q_\phi(\mathbf{z}_N|\mathbf{z}_{<N},\mathbf{x}) \tag{3}$$

Each partitioned distribution is implemented by each layer of the encoder and decoder.

[1] Audio samples and implementations are available at http://scpark20.github.io/KimchiSpeech

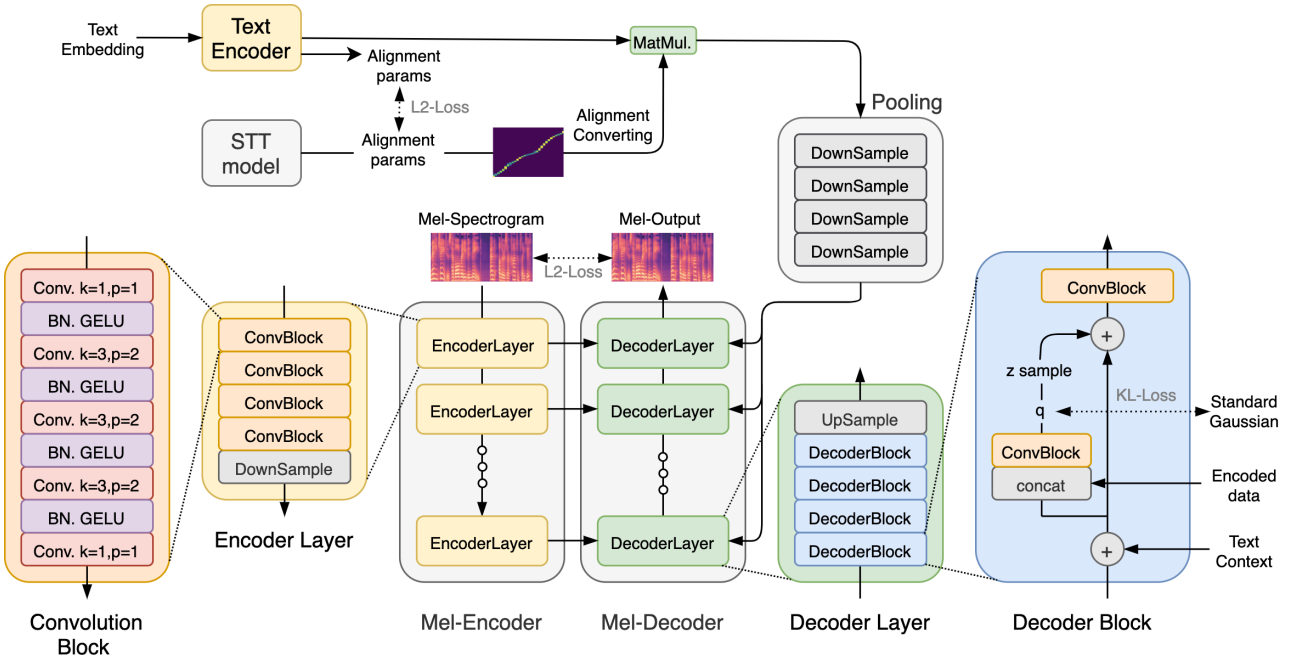

Figure 1: *KimchiSpeech Structure*

3. Model Structure

The proposed model consists of two parts: the STT model and the TTS model. The STT model uses a mel-spectrogram as the input, and outputs a text. However, this model is only used to get text-to-mel alignment in training, and is not used in inference time. The TTS model uses a text as the input, and generates a mel-spectrogram. In this study, the text-to-mel alignment obtained from STT model is used to get a mel-to-text alignment.

3.1. Speech-to-Text

The STT model is similar in structure to the Tacotron2 [2]. However, the input and output of the model are interchanged, and cross entropy is used as a loss function. The encoder of Tacotron2 functions as a mel-encoder, and the decoder functions as a text-decoder.

In [17, 18], Gaussian mixture model-based attention was proposed. We simplify this method to single Gaussian attention and use it in the STT model.

For text index i and mel index j, the attention weight is calculated as a function value of the Gaussian distribution.

$$\alpha_{ij}^{STT} = \frac{1}{\sigma_i\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{j-\mu_i}{\sigma_i}\right)^2} \tag{4}$$

Intermediate parameters $(\hat{\Delta}_i, \hat{\sigma}_i)$ are obtained by applying a linear transform to the i-th hidden state s_i of the text-decoder. The trainable parameters W and b were initialized to zero.

$$(\hat{\Delta}_i, \hat{\sigma}_i) = W s_i + b \tag{5}$$

The parameters (Δ_i, σ_i) are obtained by applying the exponential function to the intermediate parameters $(\hat{\Delta}_i, \hat{\sigma}_i)$ and multiplying them by an appropriate coefficient.

$$\begin{cases} \Delta_i = c_\Delta \exp \hat{\Delta}_i \\ \sigma_i = c_\sigma \exp \hat{\sigma}_i \end{cases} \tag{6}$$

The coefficients c_Δ, c_σ were set as hyperparameters and are used to form a proper attention alignment. Experimentally, it was found that c_Δ should have a value close to the average of $mel\ length/text\ length$. Finally, the mean μ_i is obtained by adding the current delta Δ_i to the mean of the previous step μ_{i-1}.

$$\begin{cases} \mu_0 = 0 \\ \mu_i = \mu_{i-1} + \Delta_i \end{cases} \tag{7}$$

3.2. Text-to-Speech

The TTS model is constructed using the structure of the VD-VAE and one-dimensional convolutions. In training, a condition is created by using the given text and the attention alignment obtained from the STT model. The given mel-spectrogram is encoded by the mel-encoder. Thereafter, the mel-decoder output a reconstructed mel-spectrogram using the condition and encoded data.

3.2.1. Text-Encoder

The text-encoder is approximately identical to that of Tacotron2. In other words, it consists of 3 convolutions and 1 bi-directional LSTM. However, a linear layer to output attention parameters is added.

3.2.2. Convolution Block

The convolution block is a fundamental module that constructs the mel-encoder and mel-decoder. Convolutions with kernel size 1 are used in the input and output layers, and convolutions with kernel size 3 and padding size 2 are used in the middle layers. Batch normalization [19] and a GELU activation [20] follow after the all convolutions except the last layer.

3.2.3. Mel-Encoder

The mel-encoder encodes the given mel-spectrogram and the encoded data are used to get the parameters of the variational posterior in training. The mel-encoder consists of several encoder layers. Futhermore, a mel-encoder layer consists of several convolution blocks. After a mel-encoder layer, downsampling is performed to reduce the length by half by a convolution with kernel size 2 and stride 2.

3.2.4. Mel-Decoder

The mel-decoder reconstructs the mel-spectrogram from the encoded data and given condition. The mel-decoder consists of several mel-decoder layers. The number of layers is equal to the number of mel-encoder layers. Additionally, a mel-decoder layer consists of several mel-decoder blocks. After a mel-decoder layer, up-sampling is performed to double the length by a transposed convolution with kernel size 2 and stride 2.

The decoder block performs significant roles in the hierarchical VAE structure. in training, the variational posterior is calculated using the encoded data, the given condition, and the hidden vector output from the lower layer. A $\mathbf{z}$ vector is sampled from the variational posterior and this is added to the hidden vector. Then the convolution block is performed and the result is finally output. In the inference time, a $\mathbf{z}$ vector is sampled from the prior, and not the variational posterior.

In this research, we aimed to simplify the decoding process for fast inference. Unlike previous studies [10, 13, 16], we set the prior to the standard Gaussian distribution. Therefore, additional operations are not required to obtain a prior. In addition, the $\mathbf{z}$ sample is simply added to the subvector of the hidden vector. Therefore, no additional operations are performed to match the dimensions of the $\mathbf{z}$ sample to the dimension of the hidden vector.

3.2.5. STT-aided Gaussian Attention

The text-to-mel alignment obtained from the STT model should be converted into a mel-to-text alignment for use in the TTS model. To this end, first, the logarithm of the each element of the alignment matrix α^{STT} (see eq. 4) is taken.

$$L_{ji}^{TTS} = \log(\alpha_{ij}^{STT} + \epsilon) \tag{8}$$

The ϵ is a small value, and is set to 1e-8 in the implementation. Then, take the softmax function along the text axis.

$$\alpha_{ji}^{TTS} = \frac{\exp(L_{ji}^{TTS})}{\sum_{i'} \exp(L_{ji'}^{TTS})} \tag{9}$$

In inference time, a text-to-mel alignment can be obtained using alignment parameters output from the text-encoder in TTS model and is converted into a mel-to-text alignment by the above process.

The length of the mel-spectrogram cannot be determined using the alignment parameters. To address this problem, an end-of-sequence (EOS) token is appended to the end of the text. Thereafter, in the alignment parameters output from the STT model, modify the value of delta in the index of EOS token Δ_{EOS} as follows.

$$\Delta_{EOS} = T - \sum_{i}^{L} \Delta_i \tag{10}$$

T is the length of the mel-spectrogram, and L is the length of the text. The text-encoder is trained to output this value. In inference time, the deltas are summed cumulatively, and the mean value in the index of the EOS token is regarded as the length of the mel-spectrogram.

4. Details

4.1. Training Details

Texts used for training are normalized and converted to phonemes using the grapheme-to-phoneme (G2P) library [21]. This helps to make the pronunciation correct. The Adam optimizer [22] with a learning rate of 1e-4 and a weight decay of 1e-6 was used and the model was trained in 800k steps with a batch size of 32.

The total loss function $\mathcal{L}_{total}$ consists of the cross-entropy loss $\mathcal{L}_{CE}$ of the STT model, reconstruction loss $\mathcal{L}_{recon}$, KL-divergence loss $\mathcal{L}_{KL}$ and alignment parameter loss $\mathcal{L}_{align}$ of the TTS model.

$$\mathcal{L}_{total} = \mathcal{L}_{recon} + \beta\mathcal{L}_{KL} + \mathcal{L}_{CE} + \mathcal{L}_{align} \tag{11}$$

To prevent posterior collapse, the coefficient of the KL-divergence loss is increased linearly from 0.0 to 1.0 over 50k steps. Although we used other annealing methods [23], there was no significant difference if training for a long time.

4.2. Inference Details

From [8, 9], a stable output was generated by reducing the temperature that is a coefficient multiplied by the standard deviation of the prior. In this study, we generated a stable output by using a truncated normal distribution. The experiment is described in section 6.2.

In existing TTS models [3, 5], the length of the mel-frames per text token is limited to an integer using hard attention. However, in KimchiSpeech, using Gaussian attention, the length can be a float value. Therefore, it is possible to precisely control the speed such as 1.01 and 1.02 times.

4.3. Model Configurations

We provide two model configurations, KimchiSpeech-W and KimchiSpeech-S. KimchiSpeech-W is a weight-optimized light version, and KimchiSpeech-S is a rapid version that is optimized for GPU computation. For a small size, it is advantageous to perform low-dimensional operations several times. Therefore, KimchiSpeech-W has several blocks per layer with a small hidden dimension. On the other hand, in order to run efficiently on a GPU, KimchiSpeech-S has only one block per layer with a large hidden dimension.

Table 1: *Model Configurations*

Hyperparams	**KimchiSpeech-W**	**KimchiSpeech-S**
Embedding Dim	128	128
Text Encoder Dim	128	128
Number of Layers	4	5
Number of Blocks	4	1
Hidden Dim	128	512
z Dim	16	16

Table 2: *Model Comparison*

Model	MOS 95%C.I.	RTF (GPU)	RTF (CPU)	Params
GT	4.28±0.14	-	-	-
Tacotron2	3.29±0.09	8.7x	2.2x	28.2M
FastSpeech	3.49±0.05	187x	32x	23M
FastSpeech2	3.57±0.07	205x	43x	27M
KimchiSpeech-W	**3.59±0.07**	238x	**109x**	**3.3M**(31M)
KimchiSpeech-S	3.54±0.08	**470x**	83x	18.6M(70M)

5. Experiments

LJspeech-1.1 [24] was used as training and test set for the experiment. 523 samples of LJ001 and LJ002 were used as the test set, and 349 samples of LJ003 were used as the validation set. The MOS test was conducted for English native speakers through Amazon MTurk service. 50 samples were randomly extracted from the test set. For the prior, as shown in Section 6.2, a truncated normal distribution with a range of (-1, 1) is used. To convert the mel-spectrogram into a waveform, the Parallel WaveGAN [25] implemented by Kan Bayashi[2] was used and the pretrained model name is *ljspeech_parallel_wavegan.v3*. For Tacotron2, FastSpeech and FastSpeech2, the pretrained models in espnet [26] were used. Speed measurements were recorded from the AMD Ryzen 7 2700x and NVIDIA RTX 2080 Ti.

5.1. Audio Quality

From Table 2, KimchiSpeech has a similar or superior quality to existing TTS models with state-of-the-art quality.

5.2. Model Weight & Inference Speed

From Table 2, the number outside the parentheses is the number of parameters required for inference. That is, the number of parameters of the text-encoder and mel-decoder are included. The STT model, mel-encoder, and convolution blocks for the posterior are excluded. The number of parameters of the overall model required for training is written in parentheses.

We measured the time from the input of text to the output of the mel-spectrogram and calculated its duration to compare to the duration of the speech. The two models of KimchiSpeech are faster and have a smaller size compared to existing models. KimchiSpeech-W has the smallest model size and is particularly optimized for CPU computation. Although KimchiSpeech-S has a larger model size than the KimchiSpeech-W, it is optimized for GPU computation.

6. Ablation Study

Factors considered to determine the model are as follows.

- **1. Number of layers in encoder and decoder**
- **2. Range of truncated normal distributions**
- 3. Dimensions of the encoder and decoder
- 4. Configuration of convolution block

Factors 3 and 4 can be determined by the measuring the reconstruction loss at the beginning of training. However, factors 1 and 2 were determined by a listening test after long training.

[2] https://github.com/kan-bayashi/ParallelWaveGAN

6.1. Number of layers

Table 3 presents the MOS score and the reconstruction loss after training up to 400k. In KimchiSpeech-W, the MOS score was higher for four layers than five layers. However, this difference is insignificant. In KimchiSpeech-S, the MOS score increased significantly if five layers were used. Because KimchiSpeech-S has only one convolution block per layer, the long-term context can be reflected only when a sufficient number of layers are provided. Meanwhile, the MOS score and reconstruction loss were inconsistent.

Table 3: *Test for number of layers*

Model	Layers	MOS 95%C.I.	Recon. Loss
KimchiSpeech-W	5	3.83±0.13	**0.0424**
KimchiSpeech-W	4	**3.87±0.12**	0.0431
KimchiSpeech-S	5	**3.87±0.12**	0.0579
KimchiSpeech-S	4	3.53±0.15	**0.0543**

6.2. Truncated Normal Distribution

Table 4 presents values that must be truncated to produce good quality speech. We compared the outputs generated by sampling $\mathbf{z}$ from the normal and truncated normal distributions. Consequently, the outputs from the normal distribution were the unsatisfactory, and the best obtained was from the truncated normal distribution in the range of (-1, 1).

Table 4: *Test for sampling*

Model	Range	MOS 95%C.I.
KimchiSpeech-S	(-1, 1)	**3.90±0.09**
KimchiSpeech-S	(-2, 2)	3.78±0.10
KimchiSpeech-S	(-3, 3)	3.77±0.09
KimchiSpeech-S	not truncated	3.76±0.10

7. Conclusion

We proposed a TTS model using hierarchical VAE and STT-aided attention through KimchiSpeech. Using the hierarchical VAE, a high-quality mel-spectrogram with various prosody can be generated, and using the STT-aided attention, a attention alignment can be obtained independently of the TTS model. In addition, the length of the genearated speech can be flexibly controlled using Gaussian attentions.

We also provided two configurations, KimchiSpeech-W and KimchiSpeech-S. The former is a weight-optimized light version and the latter is a speed-optimized fast version. These two models are used differently, depending on their purpose.

Compared to the previously proposed state-of-the-art models, KimchiSpeech can generate outputs with similar or superior quality. However, it is difficult to manage the quality of real human voices. We will address these limitations in subsequent researches.

8. References

[1] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio *et al.*,

"Tacotron: Towards end-to-end speech synthesis," *arXiv preprint arXiv:1703.10135*, 2017.

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

[3] 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 *Advances in Neural Information Processing Systems*, 2019, pp. 3165–3174.

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[5] Y. Ren, C. Hu, T. Qin, S. Zhao, Z. Zhao, and T.-Y. Liu, "Fastspeech 2: Fast and high-quality end-to-end text-to-speech," *arXiv preprint arXiv:2006.04558*, 2020.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[7] C. Miao, S. Liang, M. Chen, J. Ma, S. Wang, and J. Xiao, "Flow-tts: A non-autoregressive network for text to speech based on flow," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7209–7213.

[8] J. Kim, S. Kim, J. Kong, and S. Yoon, "Glow-tts: A generative flow for text-to-speech via monotonic alignment search," *arXiv preprint arXiv:2005.11129*, 2020.

[9] Y. Lee, J. Shin, and K. Jung, "Bidirectional variational inference for non-autoregressive text-to-speech," in *International Conference on Learning Representations*, 2020.

[10] P. Liu, Y. Cao, S. Liu, N. Hu, G. Li, C. Weng, and D. Su, "Vara-tts: Non-autoregressive text-to-speech synthesis based on very deep vae with residual attention," *arXiv preprint arXiv:2102.06431*, 2021.

[11] M. Bińkowski, 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.

[12] J. Donahue, S. Dieleman, M. Bińkowski, E. Elsen, and K. Simonyan, "End-to-end adversarial text-to-speech," *arXiv preprint arXiv:2006.03575*, 2020.

[13] R. Child, "Very deep vaes generalize autoregressive models and can outperform them on images," *arXiv preprint arXiv:2011.10650*, 2020.

[14] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[15] C. K. Sønderby, T. Raiko, L. Maaløe, S. K. Sønderby, and O. Winther, "Ladder variational autoencoders," *arXiv preprint arXiv:1602.02282*, 2016.

[16] A. Vahdat and J. Kautz, "Nvae: A deep hierarchical variational autoencoder," *arXiv preprint arXiv:2007.03898*, 2020.

[17] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.

[18] E. Battenberg, R. Skerry-Ryan, S. Mariooryad, D. Stanton, D. Kao, M. Shannon, and T. Bagby, "Location-relative attention mechanisms for robust long-form speech synthesis," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6194–6198.

[19] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.

[20] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

[21] J. Park, Kyubyong Kim, "g2pe," https://github.com/Kyubyong/g2p, 2019.

[22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[23] H. Fu, C. Li, X. Liu, J. Gao, A. Celikyilmaz, and L. Carin, "Cyclical annealing schedule: A simple approach to mitigating kl vanishing," *arXiv preprint arXiv:1903.10145*, 2019.

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

[25] R. Yamamoto, E. Song, and J.-M. Kim, "Parallel wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6199–6203.

[26] T. Hayashi, R. Yamamoto, K. Inoue, T. Yoshimura, S. Watanabe, T. Toda, K. Takeda, Y. Zhang, and X. Tan, "Espnet-TTS: Unified, reproducible, and integratable open source end-to-end text-to-speech toolkit," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7654–7658.