1. INTRODUCTION

Modern text-to-speech (TTS) synthesis systems, using deep neural networks trained on large quantities of data, are able to generate natural speech. TTS is a multimodal generation problem, since a given text input can be realized in many ways, both in terms of gross structure, e.g., with different prosody or pronunciation, and in low level signal detail, e.g., with different, possibly perceptually similar, structure, e.g., with different prosody or pronunciation, and in low quantities. Nevertheless, we find that the model is able to align the text and generate high fidelity speech without framing artifacts.

End-to-end generation of waveform samples from text has been elusive, due to the difficulty of efficiently modeling strong temporal dependencies in the waveform, e.g., phase, which must be consistent over time. Sample-level autoregressive vocoders handle such dependencies by conditioning generation of each waveform sample on all previously generated samples. Due to their highly sequential nature, they are inefficient to sample from on modern parallel hardware.

In this paper we integrate a flow-based vocoder into a Tacotron-like frame-level autoregressive model of speech waveforms, conditioned on a sequence of characters or phones. The sequence-to-sequence decoder attends to the input text, and produces conditioning features for a normalizing flow which generates waveform frames.

The output waveform is modeled as a sequence of fixed-length frames, each containing hundreds of samples. Dependencies between samples within each frame are modeled using a normalizing flow, enabling parallel training and synthesis. Longer-term dependencies are modeled autoregressively by conditioning on previous frames. The resulting model solves a more difficult task than text-to-spectrogram generation since it must synthesize the fine-time structure in the waveform (ensuring that edges of successive output frames line up correctly) as well as produce coherent long-term structure (e.g., prosody and semantics). Nevertheless, we find that the model is able to align the text and generate high fidelity speech without fringing artifacts.

Recent work has integrated normalizing flows with sequence-to-sequence TTS models, to enable a non-autoregressive decoder [15], or improve modelling of the intermediate mel spectrogram [16, 17]. [18] and [19] also propose end-to-end TTS models which directly generate waveforms, but rely on spectral losses and mel spectrograms for alignment. In contrast, we avoid spectrogram generation altogether and use a normalizing flow to directly model the waveform, in a fully end-to-end approach, simply maximizing likelihood.

2. WAVE-TACOTRON MODEL

Wave-Tacotron extends the Tacotron attention-based sequence-to-sequence model, generating frames of non-overlapping waveform samples instead of spectrogram frames. See Figure 1 for an overview. The encoder is a CBHG [11], which encodes a sequence of $I$ token (characters or phones) embeddings $x_{1:I}$. The text encoding is passed to a block-autoregressive decoder using attention, producing conditioning features $c_t$ for each output step $t$. A normalizing flow $g$ uses these features to sample an output waveform for that step, $y_t$:

$$e_{1:I} = \text{encode}(x_{1:I})$$

$$c_t = \text{decode}(e_{1:t}, y_{1:t-1})$$

$$y_t = g(z_t; c_t) \text{ where } z_t \sim \mathcal{N}(0, I)$$

The flow $g$ converts a random vector $z_t$ sampled from a spherical Gaussian into a waveform frame. $c_t$ is also passed to a simple linear
We replace ReLU activations with tanh in the pre-net. Since we \( \sigma \) where is a sigmoid nonlinearity. Each output \( y_t \in \mathbb{R}^{K} \) by default comprises 40 ms of speech: \( K = 900 \) at a 24 kHz sample rate. The setting of \( K \) controls the trade-off between parallelizability through normalizing flows, and sample quality through autoregression. The network structure follows [12], with minor modifications to the decoder. We use location-sensitive attention [21], which was more stable than the non-content-based GMM attention from [22]. We replace ReLU activations with tanh in the pre-net. Since we sample from the normalizing flow in the decoder loop, we do not need to apply pre-net dropout when sampling as in [2]. We also do not use a post-net [2]. Waveform frames generated at each decoder step are simply concatenated to form the final signal. Finally, we add a skip connection over the decoder pre-net and attention layers, to give the flow direct access to the samples directly preceding the current frame. This is essential to avoid discontinuities at frame boundaries.

Similar to [11], the size \( K \) of \( y_t \) is controlled by a reduction factor \( R \), where \( K = 320 \cdot R \), and \( R = 3 \) by default. Like [11], the autoregressive input to the decoder consists of only the final \( K/R \) samples of the output from the previous step \( y_{t-1} \).

### 2.1. Flow

The normalizing flow \( g(z_t; c_t) \) is a composition of invertible transformations which maps a noise sample drawn from a spherical Gaussian to a waveform segment. Its representational power comes from composing many simple functions. During training, the inverse \( g^{-1} \) maps the target waveform to a point under the spherical Gaussian whose density is easy to compute.

Since training and generation require efficient density computation and sampling, respectively, \( g \) is constructed using affine coupling layers [23]. Our normalizing flow is a one-dimensional variant of Glow [24], similar to [8]. The input \( z_t \in \mathbb{R}^{K} \) (or \( y_t \) for \( g^{-1} \)) is first squeezed into a sequence of \( J \) frames, each with dimension \( L = 10 \). The flow is divided into \( M \) stages, each operating at a different temporal resolution, following the multiscale configuration described in [25]. This is implemented by interleaving squeeze operations which reshape the sequence between each stage, halving the number of timesteps and doubling the dimension, as illustrated in Figure 2.

Each stage is further composed of \( N \) flow steps as shown in the top of Figure 1. We do not use data-dependent initialization for our ActNorm layers as in [24]. The affine coupling layers emit transformation parameters using a 3-layer 1D convnet with kernel sizes 3, 1, 3, with 256 channels. Through the use of squeeze layers, coupling layers in different stages use the same kernel widths yet have different effective receptive fields relative to \( y_t \). The default configuration uses \( M = 5 \) and \( N = 12 \), for a total of 60 steps.

The conditioning signal \( c_t \) consists of a single vector for each decoder step, which must encode the structure of \( y_t \) across hundreds of samples. Since the flow internally treats \( z_t \) and \( y_t \) as a sequence of \( J \) frames, we find it helpful to upsample \( c_t \) to match this framing. Specifically, we replicate \( c_t \) \( L \) times and concatenate sinusoidal position embeddings to each time step, similar to [26], albeit with linear spacing between frequencies. The upsampled conditioning features are appended to the inputs at each coupling layer.

To avoid issues associated with fitting a real-valued distribution to discrete waveform samples, we adapt the approach in [27] by quantizing samples into \([0, 2^{16} - 1]\) levels, dequantizing by adding uniform noise in \([0, 1]\), and rescaling back to \([-1, 1]\). We also found it helpful to pre-emphasize the waveform, whitening the signal spectrum using an FIR highpass filter with a zero at 0.9. The generated signal is de-emphasized at the output of the model. This pre-emphasis can be thought of as a flow without any learnable parameters, acting as an inductive bias to place higher weight on high frequencies. Although not critical, we found that this improved subjective listening test results.

As is common practice with generative flows, we find it helpful to reduce the temperature of the prior distribution when sampling. This amounts to simply scaling the prior covariance of \( z_t \), in equation 2 by a factor \( T^2 < 1 \). We find \( T = 0.7 \) to be a good default setting.

During training, the inverse of the flow, \( g^{-1} \), maps the target waveform \( y_t \) to the corresponding \( z_t \). Conditioning features \( c_t \) are calculated using teacher-forcing of the sequence-to-sequence network. The negative log-likelihood objective of the flow at step \( t \) is [23]:

\[
\mathcal{L}_{\text{flow}}(y_t) = -\log p(y_t|z_{1:t}, y_{1:t-1}) = -\log p(y_t|c_t) = -\log N(z_t; 0, I) - \log |\det(\partial c_t/\partial y_t)|
\]  

where \( z_t = g^{-1}(y_t; c_t) \). A binary cross entropy loss is also used on
the stop token classifier:

\[
L_{\text{os}}(s_t) = -\log p(s_t|x_{1:t}, y_{1:t-1}) = -\log p(s_t|x_t)
\]

\[
= -s_t \log \hat{s}_t - (1 - s_t) \log(1 - \hat{s}_t)
\]

(6)

where \(s_t\) is the ground truth stop token label, indicating whether \(t\) is the final step in the utterance. In practice we zero-pad the signal with several frames labeled with \(s_t = 1\) in order to provide a better balanced training signal to the classifier. We use the mean of eq. 5 and 6 across all decoder steps \(t\) as the overall loss.

### 2.2. Flow vocoder

As a baseline, we experiment with a similar flow network in a fully feed-forward vocoder context, which generates waveforms from mel spectrograms as in [8,9]. We follow the architecture in Fig. 2 with flow frame length \(L = 15\), \(M = 6\) stages and \(N = 10\) steps per stage. Input mel spectrogram features are encoded using a 5-layer dilated convnet conditioning stack using 512 channels per layer, similar to that in [3]. Features are upsampled to match the flow frame rate of 1600 Hz, and concatenated with sinusoidal embeddings to indicate position within each feature frame, as above. This model is highly parallelizable, since it does not make use of autoregression.

### 3. EXPERIMENTS

We experiment with two single-speaker datasets: A proprietary dataset containing about 39 hours of speech, sampled at 24 kHz, from a professional female voice talent which was used in previous studies [4,11,10]. In addition, we experiment with the public LJ speech dataset [20] of audiobook recordings, using the same splits as [22]: 22 hours for training and a held-out 130 utterance subset for evaluation. We upsample the 22.05 kHz audio to 24 kHz for consistency with the first dataset. Following common practice, we use a text normalization upsample the 22.05 kHz audio to 24 kHz for consistency with the first dataset. Following common practice, we use a text normalization

- **3.1. Tacotron models** We train a Tacotron model with a post-net (Tacotron-PN), consisting of a 20-layer non-causal WaveNet stack split into two dilation cycles. This converts mel spectrograms output by the decoder to full linear frequency spectrograms which are inverted to waveform samples using 100 iterations of the Griffin-Lim algorithm [29], similar to [11].

- **3.2. Wave-Tacotron models** are trained using the Adam optimizer for 500k steps across 32 Google TPU v3 cores, with batch size 256 for the proprietary dataset and 128 for LJ, whose average utterance duration is longer.

### 3.3. Objective metrics

- **Mel cepstral distortion** (MCD), the root mean squared error against the ground truth signal, computed on 13 dimensional MFCCs [30] using dynamic time warping [31] to align features computed on the synthesized audio to those computed from the ground truth. MFCC features are computed from an 80-channel log-mel spectrogram using a 50ms Hann window and hop of 12.5ms. (2) Mel spectral distortion (MSD), which is the same as MCD but applied to the log-mel spectrogram magnitude instead of cepstral coefficients. This captures harmonic content which is explicitly removed in MCD due to liftering. (3) Character error rate (CER) computed after transcribing the generated speech with the Google speech API, as in [22]. This is a rough measure of intelligibility, invariant to some types of acoustic distortion. In particular CER is sensitive to robustness errors caused by stop token or attention failures. Lower values are better for all metrics except MOS. As can be seen below, although MSD and MCD tend to negatively correlate with subjective MOS, we primarily focus on MOS.

### Table 1

| Model               | Vocoder          | Input  | MCD  | MSD  | CER   | MOS   |
|---------------------|------------------|--------|------|------|-------|-------|
| Ground truth        | –                 | –      | 8.7  | 4.56 | ±0.04 |
| Tacotron-PN         | Griffin-Lim char | 5.87   | 9.0  | 3.68 | ±0.08 |
| Tacotron-PN         | Griffin-Lim phone| 5.83   | 8.6  | 3.74 | ±0.07 |
| Tacotron            | WaveRNN char     | 4.32   | 9.1  | 4.36 | ±0.05 |
| Tacotron            | WaveRNN phone    | 4.31   | 8.8  | 3.47 | ±0.04 |
| Tacotron            | Flowcoder char   | 4.22   | 11.3 | 3.34 | ±0.07 |
| Tacotron            | Flowcoder phone  | 4.15   | 10.6 | 3.31 | ±0.07 |
| Wave-Tacotron –    | char              | 4.84   | 9.4  | 4.07 | ±0.06 |
| Wave-Tacotron –    | phone             | 4.64   | 9.2  | 4.23 | ±0.06 |

1We include samples from an unconditional Wave-Tacotron, removing the

### Table 1. TTS performance on the proprietary single speaker dataset.

Comparing MOS across different systems, in all cases Wave-Tacotron performs substantially better than the Griffin-Lim baseline, but not as well as Tacotron + WaveRNN. Using the flow as a vocoder results in MCD and MSD on par with Tacotron + WaveRNN, but increased CER and low MOS, where raters noted robotic and gravely sound quality. The generated audio contains artifacts on shorter timescales, which have little impact on spectral metrics computed with 50ms windows. In contrast, Wave-Tacotron has higher MCD and MSD than the Tacotron + WaveRNN baseline, but similar CER and only slightly lower MOS. The significantly improved fidelity of Wave-Tacotron over Flowcoder indicates the utility of autoregressive conditioning, which resolves some uncertainty about the fine-time structure within the signal that Flowcoder must implicitly learn to model. However, fidelity still falls short of the WaveRNN baseline.
work, processing the full spectrogram generated by Tacotron in parallel. Wave-Tacotron on TPU, and about twice as fast as Tacotron-PN using 1000 Griffin-Lim iterations. On its own, spectrogram generation using Tacotron is quite efficient, as demonstrated by the fast speed of Tacotron-PN with 100 Griffin-Lim iterations. On parallel hardware, and with sufficiently large decoder steps, the additional overhead in the decoder loop of the Wave-Tacotron flow, essentially a very deep convnet, is small over the fully parallel Flowcoder.

3.2. Ablations
Finally, Table 4 compares several Wave-Tacotron variations, using a shallower 2-layer decoder LSTM. As shown in the top, the optimal sampling temperature is $T = 0.7$. Removing pre-emphasis, position embeddings in the conditioning stack, or the decoder skip connection, all hurt performance. Removing the skip connection is particularly deleterious, leading to clicking artifacts caused by discontinuities at frame boundaries which significantly increase MCD, MSD, and CER, to the point where we chose not to run listening tests.

Decreasing the size of the flow network by halving the number of coupling layer channels to 128, or decreasing the number of stages $M$ or steps per stage $N$, all significantly reduce quality in terms of MOS. Finally, we find that reducing $R$ does not hurt MOS, although it slows down generation (see Table 3) by increasing the number of autoregressive decoder steps needed to generate the same audio duration. Increasing $R$ to 4 significantly reduces MOS due to poor prosody, with raters commenting on the unnaturally fast speaking rate. This suggests a trade-off between parallel generation (by increasing $K$, decreasing the number of decoder steps) and naturalness. Feeding back more context in the autoregressive decoder input or increasing the capacity of the decoder LSTM stack would likely mitigate these issues; we leave such exploration for future work.

4. DISCUSSION
We have proposed a model for end-to-end (normalized) text-to-speech waveform synthesis, incorporating a normalizing flow into the autoregressive Tacotron decoder loop. Wave-Tacotron is able to directly generate high quality speech waveforms, conditioned on text, using a single model without a separate vocoder. Training does not require complicated losses on hand-designed spectrogram or other mid-level features, but simply maximizes the likelihood of the training data. The hybrid model structure combines the simplicity of attention-based term prosody modeling, leads to more efficient models in practice. Exploring the feasibility of adapting the proposed model to fully parallel TTS generation remains an interesting direction for future work.

Although the network structure exposes the output frame size $K$ as a hyperparameter, the decoder remains fundamentally autoregressive, requiring sequential generation of output frames. This puts the approach at a disadvantage compared to recent advances in parallel TTS, unless the output step size can be made very large. Exploring the feasibility of adapting the proposed model to fully parallel TTS generation remains an interesting direction for future work. It is possible that the separation of concerns inherent in the two-step factorization of the TTS task, i.e., separating responsibility for high fidelity waveform generation from text-alignment and longer term prosody modeling, leads to more efficient models in practice. Separate training of the two models also allows for the possibility of training them on different data, e.g., training a vocoder on a larger corpus of untranscribed speech. Finally, it would be interesting to explore more efficient alternatives to flows in a similar text-to-waveform setting, e.g., diffusion probabilistic models, or GANs, which can be more easily optimized in a mode-seeking fashion that is likely to be more efficient than modeling the full data distribution.

---

### Table 2. TTS performance on LJ Speech with character inputs.

| Model            | Vocoder       | Input   | MCD    | MSD     | CER     | MOS    |
|------------------|---------------|---------|--------|---------|---------|--------|
| Ground truth     | –             | –       | 7.8    | 4.51    | ±0.05   | –      |
| Tacotron-PN      | Griffin-Lim   | char    | 7.20   | 12.39   | 7.4     | 3.26   |
| Tacotron         | WaveRNN       | char    | 6.69   | 12.23   | 6.1     | 4.47   |
| Tacotron         | Flowcoder     | char    | 5.80   | 11.61   | 13.0    | 3.11   |
| Wave-Tacotron    | –             | char    | 6.87   | 11.44   | 9.2     | 3.56   |

### Table 3. Generation speed in seconds, comparing one TPU v3 core to a 6-core Intel Xeon W-2135 CPU, generating 5 seconds of speech conditioned on 90 input tokens, batch size 1. Average of 500 trials.

| Model            | $R$ | Vocoder       | TPU   | CPU   |
|------------------|-----|---------------|-------|-------|
| Tacotron-PN      | 2   | Griffin-Lim, 100 iterations | 0.14  | 0.88  |
| Tacotron-PN      | 2   | Griffin-Lim, 1000 iterations | 1.11  | 7.71  |
| Tacotron         | 2   | WaveRNN       | 5.34  | 63.38 |
| Tacotron         | 2   | Flowcoder     | 0.49  | 0.97  |
| Wave-Tacotron    | 1   | –             | 0.80  | 5.26  |
| Wave-Tacotron    | 2   | –             | 0.64  | 3.25  |
| Wave-Tacotron    | 3   | –             | 0.58  | 2.52  |
| Wave-Tacotron    | 4   | –             | 0.55  | 2.26  |

### Table 4. Ablations on the proprietary dataset using phone inputs and a shallow decoder residual LSTM stack of 2 layers with 256 units. Unless otherwise specified, samples are generated using $T = 0.8$.

| Model            | $R$ | $M$ | $N$ | MCD    | MSD    | CER    | MOS    |
|------------------|-----|-----|-----|--------|--------|--------|--------|
| Base $T = 0.8$   | 5   | 12  | 6.43 | 9.20   | 9.0    | 4.01   |
| $T = 0.6$        | 3   | 12  | 5.02 | 9.0    | 5.1    | 4.12   |
| $T = 0.7$        | 3   | 12  | 4.71 | 9.2    | 4.2    | 4.16   |
| $T = 0.9$        | 3   | 12  | 4.34 | 9.0    | 3.7    | 3.77   |
| no pre-emphasis  | 3   | 12  | 4.46 | 9.0    | 3.8    | 3.85   |
| no position emb. | 3   | 12  | 4.64 | 9.0    | 3.7    | 3.70   |
| no skip connection | 3   | 12  | 4.59 | 9.0    | 3.9    | 3.70   |
| 128 flow channels | 3   | 12  | 4.00 | 9.0    | 3.7    | 3.70   |
| 30 steps, 5 stages | 3   | 6   | 4.33 | 9.0    | 3.7    | 3.70   |
| 60 steps, 4 stages | 3   | 15  | 4.40 | 9.0    | 3.7    | 3.70   |
| 60 steps, 3 stages | 3   | 20  | 4.50 | 9.0    | 3.9    | 3.70   |
| $K = 320$        | 1   | 5   | 3.50 | 8.8    | 3.5    | 3.50   |
| $K = 640$        | 2   | 5   | 4.50 | 9.0    | 3.5    | 3.50   |
| $K = 1280$       | 4   | 5   | 4.40 | 9.0    | 3.5    | 3.50   |

---

---

---
5. ACKNOWLEDGEMENTS

The authors thank Jenelle Feather for initial work integrating a flow into Tacotron, Rif A. Saurous for facilitating the genesis of this work, and Tom Bagby, David Kao, Matt Shannon, Daisy Stanton, Yonghui Wu, Heiga Zen, and Yu Zhang for helpful discussions and feedback.

6. 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,” in Proc. Interspeech, 2017.

[2] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, et al., “Natural TTS synthesis by conditioning WaveNet on mel spectrogram predictions,” in Proc. ICASSP, 2018, pp. 4779–4783.

[3] J. Sotelo, S. Mehri, K. Kumar, J. F. Santos, K. Kastner, A. Courville, and Y. Bengio, “Char2Wav: End-to-end speech synthesis,” in Proc. International Conference on Learning Representations, 2017.

[4] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, et al., “WaveNet: A generative model for raw audio,” CoRR abs/1609.03499, 2016.

[5] N. Kalchbrenner, E. Elsen, K. Simonyan, S. Noury, N. Casagrande, E. Lockhart, F. Stimberg, A. van den Oord, et al., “Efficient neural audio synthesis,” in Proc. International Conference on Machine Learning, 2018.

[6] A. van den Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. v. d. Driessche, E. Lockhart, L. C. Cobo, et al., “Parallel WaveNet: Fast High-Fidelity Speech Synthesis,” in Proc. International Conference on Machine Learning, 2018.

[7] W. Ping, K. Peng, and J. Chen, “ClariNet: Parallel Wave Generation in End-to-End Text-to-Speech,” in Proc. International Conference on Learning Representations, 2019.

[8] S. Kim, S.-G. Lee, J. Song, J. Kim, and S. Yoon, “FloWaveNet: A generative flow for raw audio,” in Proc. International Conference on Machine Learning, 2019.

[9] R. Prenger, R. Valle, and B. Catanzaro, “WaveGlow: A Flow-based Generative Network for Speech Synthesis,” in Proc. ICASSP, 2019.

[10] K. Kumar, R. Kumar, T. de Boissiere, L. Gestin, W. Z. Teoh, J. Sotelo, A. de Brebisson, Y. Bengio, et al., “MelGAN: Generative Adversarial Networks for Conditional Waveform Synthesis,” in Advances in Neural Information Processing Systems, 2019.

[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,” in Proc. International Conference on Learning Representations, 2020.

[12] O. Arık, H. Jun, and G. Diamos, “Fast spectrogram inversion using multi-head convolutional neural networks,” IEEE Signal Processing Letters, vol. 26, no. 1, pp. 94–98, 2018.

[13] X. Wang, S. Takaki, and J. Yamagishi, “Neural source-filter-based waveform model for statistical parametric speech synthesis,” in Proc. ICASSP, 2019, pp. 5916–5920.

[14] A. A. Gritsenko, T. Salimans, R. v. d. Berg, J. Snoek, and N. Kalchbrenner, “A Spectral Energy Distance for Parallel Speech Synthesis,” arXiv preprint arXiv:2008.01160, 2020.

[15] 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 Proc. ICASSP, 2020, pp. 7209–7213.

[16] R. Valle, K. Shih, R. Prenger, and B. Catanzaro, “Flowtron: an autoregressive flow-based generative network for text-to-speech synthesis,” arXiv preprint arXiv:2005.05957, 2020.

[17] 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.

[18] 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.

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

[20] D. Rezende and S. Mohamed, “Variational inference with normalizing flows,” in Proc. International Conference on Machine Learning, 2015, pp. 1530–1538.

[21] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, “Attention-based models for speech recognition,” in Advances in Neural Information Processing Systems, 2015.

[22] E. Bottou, R. Skerry-Ryan, S. Mariooryad, D. Stanton, D. Kao, et al., “Location-relative attention mechanisms for robust long-form speech synthesis,” in Proc. ICASSP, 2020.

[23] L. Dinh, D. Krueger, and Y. Bengio, “NICE: Non-linear independent components estimation,” in Proc. International Conference on Learning Representations, 2015.

[24] D. P. Kingma and P. Dhariwal, “Glow: Generative flow with invertible 1x1 convolutions,” in Advances in Neural Information Processing Systems, 2018, pp. 10215–10224.

[25] L. Dinh and S. Bengio, “Density estimation using Real NVP,” in Proc. International Conf. on Learning Representations, 2017.

[26] 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.

[27] B. Uria, I. Murray, and H. Larochelle, “RNADE: The real-valued neural autoregressive density-estimator,” in Advances in Neural Information Processing Systems, 2013, pp. 2175–2183.

[28] K. Ito, “The LJ Speech Dataset,” [https://keithito.com/LJ-Speech-Dataset/], 2017.

[29] D. Griffin and J. Lim, “Signal estimation from modified short-time Fourier transform,” IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 32, no. 2, 1984.

[30] R. Kubichek, “Mel-cepstral distance measure for objective speech quality assessment,” in Proc. IEEE Pacific Rim Conf. on Communications Computers and Signal Processing, 1993.

[31] D. J. Berndt and J. Clifford, “Using dynamic time warping to find patterns in time series,” in Proc. IEEE Pacific Rim Conf. on Communications Computers and Signal Processing, 1993.

[32] 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.

[33] C. Xu, H. Lu, M. Yu, C. Weng, K. Xu, P. Liu, D. Tuo, et al., “DurIAN: Duration informed attention network for multimodal synthesis,” arXiv preprint arXiv:1909.01700, 2019.

[34] 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.