INTERSPEECH 2020
October 25–29, 2020, Shanghai, China

Incremental Text to Speech for Neural Sequence-to-Sequence Models using Reinforcement Learning

Devang S Ram Mohan[1], *Raphael Lenain*[2*], *Lorenzo Foglianti*[1], *Tian Huey Teh*[1], *Marlene Staib*[1], *Alexandra Torresquintero*[1], *Jiameng Gao*[1]
[1]Papercup Technologies Ltd.
[2]Novoic
devang@papercup.com

Abstract

Modern approaches to text to speech require the entire input character sequence to be processed before any audio is synthesised. This latency limits the suitability of such models for time-sensitive tasks like simultaneous interpretation. Interleaving the action of reading a character with that of synthesising audio reduces this latency. However, the order of this sequence of interleaved actions varies across sentences, which raises the question of how the actions should be chosen. We propose a reinforcement learning based framework to train an agent to make this decision. We compare our performance against that of deterministic, rule-based systems. Our results demonstrate that our agent successfully balances the trade-off between the latency of audio generation and the quality of synthesised audio. More broadly, we show that neural sequence-to-sequence models can be adapted to run in an incremental manner.
Index Terms: text to speech, reinforcement learning

1. Introduction

Efforts towards incremental text to speech (TTS) have typically focused on more traditional, non-neural architectures [1, 2, 3]. However, advancements in neural TTS [4, 5, 6] have resulted in near human levels of naturalness and thus motivate an exploration of neural incremental TTS systems.

Neural TTS systems typically adopt sequence-to-sequence architectures which require the entire input sequence to be processed before generating any units of the output sequence. This offline characteristic is often useful; for example, a question mark at the end of a sentence would impact the intonation of preceding words. On the other hand, synthesising speech incrementally from text could be valuable. Such a model could be placed at the tail-end of an incremental speech recognition and machine translation pipeline to obtain a real-time speech-to-speech translation system.

The development of these streaming, end-to-end architectures has seen considerable attention for the tasks of automatic speech recognition [7, 8, 9, 10] and machine translation (MT) [11, 12, 13, 14]. Inspired by the approach of [12], our proposed framework develops an agent that decides whether to trigger the encoder with the next input character (i.e., *READ* in Figure 1), or trigger the decoder with the characters read thus far (i.e., *SPEAK* in Figure 1). In this manner, our approach enables us to start generating mel-spectrograms while having read only a part of the input sentence. The mapping of these mel-spectrogram frames to raw audio waveforms can be achieved with an existing neural vocoder [4, 15] by adjusting its inference behaviour.

The challenge then lies in deciding when to incorporate an additional character into this restricted input subsequence. We use the REINFORCE algorithm [16] to train an agent to make this decision.

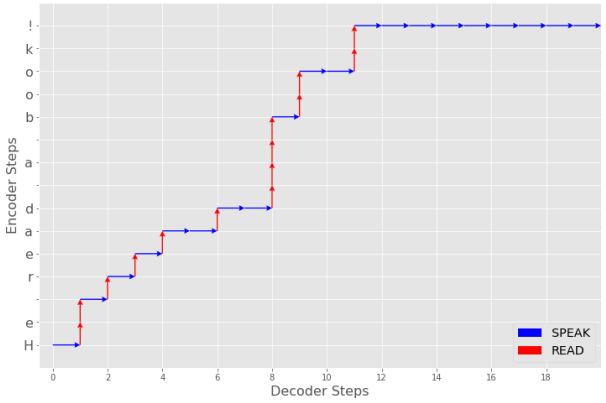


Figure 1: *Trajectory for an arbitrary sequence of READ (red, along y-axis) and SPEAK actions (blue, along x-axis)*

2. Background

[17] proposes an approach for incremental neural TTS. The model is based on the prefix-to-prefix framework [18] and leverages a policy which maintains a fixed latency (in terms of number of words) behind the input. However, it would be challenging to construct such a rule-based approach if the desired latency was to be measured in a more granular unit, such as characters or phonemes. Furthermore, a dynamic, learnt policy would allow this approach to be used for new languages and speakers without manual calibration of these parameters.

The arena of incremental machine translation has also seen advancements. [11] proposes the framework of *READ/WRITE* and once again uses rule-based policies to enable incremental machine translation. [12] models this discrete action selection task using a reinforcement learning (RL) system, which we adapt in our work. Alternatively, [13] turns this non-differentiable framework into a supervised learning problem by training a model on sequences of interleaved *READ/WRITE* decisions generated from a pre-trained model.

A major challenge in any sequence transduction task is to align the target sequence with the source at each step. [7, 8] propose methods that leverage the RNN-T model [19] to address this for the task of speech recognition. As an alternative, the approaches in [9, 10] propose architectures which utilise the fact that in speech recognition, the length of the target sequence is less than that of the source. [20, 21, 22] use encoder-decoder architectures with attention, but compute the attention alignments

[*]work done while at Papercup Technologies Ltd.

http://dx.doi.org/10.21437/Interspeech.2020-1822

in an online manner.

[23] adapts the online, monotonic attention mechanism proposed by [24] for the Tacotron 2 model. However, the motivation behind this was to ensure the surjectivity of the mapping between input elements and output frames and thus, the encoder and decoder architectures remain offline. Furthermore, the atomic input unit is a phoneme which can only be computed given the entire word. RL based approaches have also been used to generate attention weights for image captioning [25], [26, 27]. However, these attention mechanisms generate hard attention weights which is undesirable for TTS [28].

3. Tacotron 2 Modifications

Our base model builds on the Tacotron 2 model, with certain modifications for the incremental setting. Note that while these modifications may affect the quality of synthesised speech, they are necessary restrictions for incremental synthesis.

The encoder is altered by simply removing the convolutional layers and replacing the bi-directional LSTM [29, 30] with a uni-directional one. We further discard the post-net module, leaving only the attention mechanism that renders this model *offline*. Rather than modifying the computation of the alignment weights and potentially enforcing a hardness constraint, we maintain the soft attention weights and suitably restrict its scope as described in Section 4.

Finally, note that Tacotron 2 also has a vocoder component, which maps the mel-spectrogram to the raw audio waveform. We use a different vocoder architecture [15] and adapt its inference behaviour to work in a purely auto-regressive manner by restricting the number of mel-spectrogram frames input to its residual and up-sampling networks.

For the remainder of this paper, we use this modified Tacotron 2 architecture to generate mel-spectrograms with the understanding that any incremental vocoder can be leveraged for synthesis.

4. Incremental Text to Speech using Reinforcement Learning

Inspired by [12], we maintain an increasing buffer of input characters, which the model attends over to synthesise the next mel-spectrogram frame. We then train an agent to make the decision of whether to add the next input character into this buffer, or to synthesise a frame of audio based on the information in the buffer. To train this agent, we leverage the RL paradigm.

4.1. RL Setup and Notation

The RL setup consists of a decision maker, called the *agent*, interacting with an *environment*, typically over a sequence of discrete steps which we index by j. At the jth interaction step, the agent selects an action a_j, which the environment executes, and returns a new observation $\mathbf{o}_{j+1}$ (which is a representation of how its internal state has changed) and a numerical reward, r_{j+1}. In addition, the environment returns a flag which indicates whether this particular *episode* of interactions has completed, called the terminal flag. The task for the agent, then, is to learn a mapping from the space of all possible observations to a suitable action. Such a mapping, called a *policy*, should attempt to maximise the cumulative numerical reward achieved over the course of an episode (typically discounted temporally by a factor $\gamma \in [0, 1]$) [31].

Formally, let $\mathbf{x}_1, ..., \mathbf{x}_N$ denote the sequence of input character embeddings and $\mathbf{h}_1, ..., \mathbf{h}_N$ denote the corresponding encoder outputs from our modified Tacotron 2 (Section 3). Our modifications enable $\mathbf{h}_i$ to be computed without knowledge of $\mathbf{x}_{i+1}, ..., \mathbf{x}_N$. Let the associated ground-truth mel-spectrogram $\mathbf{y} \in \mathbb{R}^{128 \times T}$ consist of T frames. At the jth step of an episode, let $R(j) \in \{1, ..., N\}$ denote the number of characters that have been read and $S(j) \in \{1, ...T\}$ represent the number of audio frames generated (aligned by teacher-forcing [4] during training). Let $\alpha_{i,S(j)}$ denote the alignment weight over $\mathbf{h}_i$ while generating the $S(j)$th decoder output, $\hat{\mathbf{y}}_{S(j)}$.

Instead of using $\{\mathbf{h}_1, ..., \mathbf{h}_N\}$ to compute these weights (and thence the attention context), we use our restricted buffer $\{\mathbf{h}_1, ..., \mathbf{h}_{R(j)}\}$. This approach guarantees that, at the time of synthesising the $S(j)$th frame of audio, our Tacotron 2 model only has access to the first $R(j)$ characters.

4.2. Agent

The actions available to the agent are:

- *READ*: (step along the vertical axis in Figure 1) Provides the attention mechanism with an additional character over which it may attend.
- *SPEAK*: (step along the horizontal axis in Figure 1) Results in the generation of a mel-spectrogram frame based on the characters read thus far.

Then, a desirable learnt policy might be the agent learning to *SPEAK* as soon as there is enough *READ* context, and to resume *READ*ing only when the existing context is fully synthesised. Observe that the offline behaviour can also be obtained as a specific policy (*READ* all characters and then *SPEAK* until all frames are synthesised).

4.3. Environment

The environment uses a trained modified Tacotron 2 model to provide the agent with the requisite information and feedback.

4.3.1. Observations

Suppose we have just received action a_{j-1}. The environment increments the appropriate counter ($R(j)$ or $S(j)$, based on a_{j-1}) and passes $\mathbf{h}_1, ..., \mathbf{h}_{R(j)}$ to the attention module, which computes $\alpha_{1,S(j)}, ..., \alpha_{R(j),S(j)}$. The context vector is then

$$\mathbf{c}_{S(j)} = \sum_{i=1}^{R(j)} \alpha_{i,S(j)} \mathbf{h}_i \tag{1}$$

Since we want $\mathbf{o}_j$ to contain enough information for the agent to decide whether to *READ* or *SPEAK*, we define $\mathbf{o}_j$ to be the concatenation of:

- $\mathbf{c}_{S(j)}$: The attention context vector based on the $R(j)$ characters read thus far.
- $\alpha_{.,S(j)}[k:]$: A fixed length moving window of the latest attention weights. This term was found to be crucial for learning a good policy.
- $\mathbf{y}_{S(j)}$ (during training) or $\hat{\mathbf{y}}_{S(j)}$ (during evaluation): The most recent mel-spectrogram frame.

4.3.2. Rewards

Underpinning our RL framework is the understanding that the quality of the generated output may trade-off against the delay incurred. Thus, we define our reward as

$$r_j := r_j^D + r_j^Q \tag{2}$$

where r_j^D encourages low latency while r_j^Q encourages high quality synthesis. Motivated by the treatment in [12], we define

$$r_j^D := r_j^{CR} + r_j^{AP} \tag{3}$$

where

- r_j^{CR} is a local signal to discourage consecutive *READ* actions

$$r_j^{CR} := \omega \times (sgn(c_j - c^*) + 1) \tag{4}$$

  c_j is a counter for consecutive *READ*s, c^* is an acceptable number of consecutive *READ*s and $\omega < 0$ is a hyper-parameter.

- r_J^{AP} is a global penalty incurred only at the end of an episode

$$r_J^{AP} := \beta \times \lfloor d_T - d^* \rfloor_+ \tag{5}$$

  Geometrically, d_T corresponds to the average proportion of area under the policy path (Figure 1). A value of 1 for d_T corresponds to *READ*ing the entire input sequence before generating any output, while 0 corresponds to the unattainable scenario of synthesising all the audio without *READ*ing any characters. d^* is a target value for d_T and $\beta < 0$ is a hyper-parameter.

Prior works in MT [12, 18] have a detailed description of these terms.

To compute r_j^Q, we use the mean squared error (MSE) between the ground truth and generated mel-spectrograms (aligned using teacher forcing). While the MSE is limited as a measure of perceived quality [32], its usage as a training objective for our underlying Tacotron 2 model suggests it is suitable for our setting. We obtain a quality penalty term given by

$$r_j^Q := \lambda \times MSE(\mathbf{y}_{S(j)}, \hat{\mathbf{y}}_{S(j)}) \tag{6}$$

where $\lambda < 0$. When a *READ* is executed, r_j^Q is set to 0.

4.3.3. Terminal Flag

At train time, there are two ways that the episode can terminate:

- $R(j) = N$ (all the characters have been read) At this point the agent is forced to *SPEAK* until $S(t) = T$. It is then given a cumulative reward for these *SPEAK* actions.
- $S(j) = T$ (all the aligned mel-spectrograms have been consumed) At this point, the agent is given an additional penalty equal to the number of unread characters and the episode is terminated.

During inference, the episode runs until our Tacotron 2 model's termination criterion (i.e., the stop token) is triggered.

4.4. Agent Setup and Learning

The agent receives an observation $\mathbf{o}_j$ which is passed through a policy network consisting of a 512-dimensional GRU unit, a 2 layer dense network with ReLU non-linearity, and a softmax layer, to produce a 2-dimensional vector of action probabilities.

To learn these policy parameters θ, we use the policy gradient method [16] which maximises expected cumulative discounted reward. However, as a variance reduction technique, we replace the discounted returns G_j in the update, with a normalised advantage value [33]. To compute this we subtract a baseline return, $b_\phi(\mathbf{o}_j)$ (where ϕ parameterises a 3-layer fully connected network), and then normalise the result [33, 34]. To learn the baseline network parameters ϕ, we minimise the expected squared loss between G_j and $b_\phi(\mathbf{o}_j)$.

For both terms, the expectation is approximated by sampling a trajectory under the policy π_θ. All parameters are trained jointly on collected batches of transitions.

5. Experiments

5.1. Settings

We use the LJ Speech dataset [35], which consists of English audio from a single speaker. We partition this dataset into 12,000 train and 1,100 test/validation data points. We train our modified Tacotron 2 model for 300,000 iterations following the training routine in [4].

We set the weights of each reward component, $\omega = -1$, $\beta = -10$ and $\lambda = -100$, to ensure that the scale of contribution is comparable. The target number of consecutive characters read, c^* is set to 4 while the target average proportion of area under the policy path, d^* is set to 0.5. These values are interpretable levers that allow the model's behaviour to be tweaked. The look-back of the attention window was set to 5.

During training, actions are sampled according to the probabilities returned by the policy to encourage exploration of the observation space. While evaluating, actions are chosen greedily. We use a discount factor of 0.99 and train on batches of collected transitions at the end of every 10 episodes, using an Adam optimiser [36] initialised with a learning rate of 10^{-4}.

5.2. Benchmark Policies

To gauge the performance of our agent, we used two types of benchmark policies, inspired by [17, 12]:

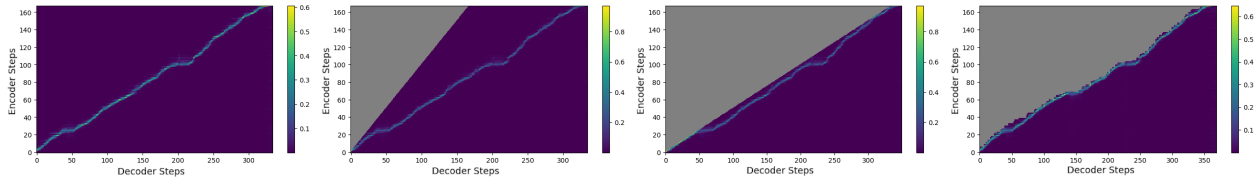
Wait-Until-End (WUE): *Execute READ actions until the text buffer is empty and then decode everything*. Since this policy has access to the entire input sentence at the time of decoding, this gives an upper bound on the quality of the synthesised speech, at the cost of the largest possible delay.

Wait-k-Steps (WkS): *Execute a READ action every k steps, and decode in between*. Despite incurring a smaller delay, the restricted access to the input sentence while decoding may impact the quality of the generated speech.

5.3. Qualitative Analysis

Figure 2 depicts the attention alignments and policy path for a sample sentence [1]. Figures 2a and 2b show that, for a large part of the decoding process, the WUE and W2S policies have access to more characters than required which highlights an avoidable latency. Figure 2c suggests that the W3S policy is able to reduce these unnecessary *READ*s. However, the resulting policy path appears to collide with the 'prominent' alignments on multiple occasions. As a result, the audio quality at these points is compromised because the decoder does not have sufficient context. This motivates the idea that an ideal policy path should *hug* the prominent alignments diagonal closely from *above* to successfully balance the quality of synthesis and latency incurred. Our learnt policy (Figure 2d) does precisely that. This suggests that the agent has in fact learnt to *READ* only when necessary and *SPEAK* only when it has something relevant to output.

[1] English (and French) audio samples can be found at https://research.papercup.com/samples/incremental-text-to-speech



(a) *Wait Until End (WUE) Policy* (b) *Wait 2 Steps (W2S) Policy* (c) *Wait 3 Steps (W3S) Policy* (d) *Learnt Policy*

Figure 2: *Policy Path with Attention Alignments (English): Each plot depicts the policy path and the attention alignments (by colour). The greyed out section represents portions of the input sentence that is unavailable as those input characters have not yet been read.*

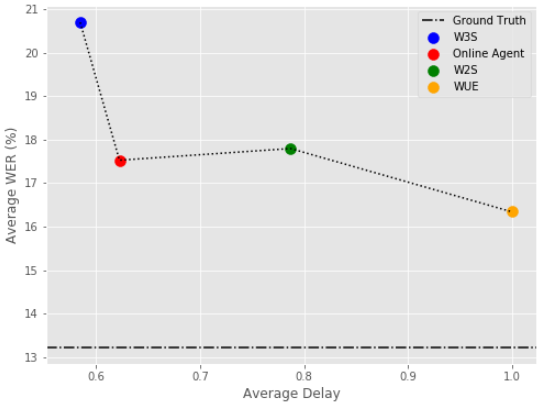


Figure 3: *Average WER vs Latency (d_T) on a test set comprising 40 samples from LJ Speech labelled by 5 annotators*

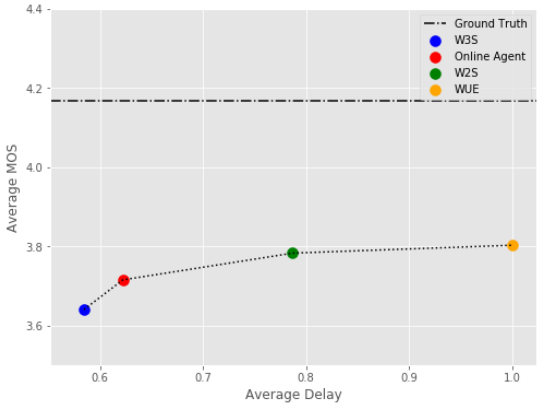


Figure 4: *Average MOS vs Latency (d_T) on a test set comprising 40 samples from LJ Speech labelled by 10 evaluators*

5.4. Quantitative Analysis

There are two aspects of the agent's performance that we track:

Quality: We compute the Mean Opinion Score (MOS) to measure the naturalness of our audio [37, 4]. We considered using a MUSHRA test [38]. However, since some policies may generate unintelligible samples of audio, which in turn could be scored below a noisy anchor, this approach was set aside. We are also interested in measuring the intelligibility of the synthesised speech. Automatic speech recognition systems use *word-error rate* (WER) to measure the transcription quality [39]. Following this approach, we obtain human transcriptions of the speech and compute the WER against the ground truth.

Latency: We use the proportion of area under the policy path, $d_T \in [0, 1]$ described in Section 4.3. This metric lacks interpretability in terms of the actual delay incurred (e.g. the number of extra characters read). An alternate average lagging metric has been proposed in the MT setting [18]. However, the skewed ratio between the source and target lengths for TTS coupled with a soft alignment between source and target make this metric challenging to adapt to TTS.

5.4.1. Results

Figures 3 and 4 depict the inherent trade-off between quality and latency. The ground truth marker depicts the value of the relevant metric for the vocoded ground truth mel-spectrograms.

We begin by observing that the W3S policy incurs the least delay, closely followed by our online agent, while the W2S and WUE policies incur substantial delays. In terms of intelligibility, our online agent achieves a better WER than W3S, and even outperforms W2S despite its sizeable latency advantage. In terms of naturalness, our agent similarly outperforms W3S on MOS, but in this case, W2S was, as expected, able to leverage the additional latency to produce more natural sounding speech.

These findings establish that our agent is able to learn a policy that successfully balances the quality of the synthesised output against the latency incurred. The W2S policy is either comparable (intelligibility) or marginally better (naturalness) than our online agent, but in doing so, performs a large number of premature READ actions. Our agent incurs a slightly larger delay than the W3S policy, and manages to outperform it on all quality metrics.

6. Future Work

Our results show that for neural sequence-to-sequence, attention-based TTS models, there is no algorithmic barrier to incrementally synthesising speech from text. It is also interesting to analyse the learnt policy for different languages given the varied challenges posed (eg. elisions and liasons in French [40]). We provide samples from an agent trained on the French SIWIS dataset [41] with the same setup as described, on our samples page[1].

Furthermore, we used a modified Tacotron 2 model, pre-trained on full sentences. It would be interesting to analyse whether jointly learning the Tacotron weights helps synthesise partial fragments of a sentence better.

7. Acknowledgements

We would like to thank Simon King, Mark Herbster and Mark Gales for their valuable input on this research.

8. References

[1] T. Baumann and D. Schlangen, "The inprotk 2012 release," in *NAACL-HLT Workshop on Future Directions and Needs in the Spoken Dialog Community: Tools and Data*. Association for Computational Linguistics, 2012, pp. 29–32.

[2] ——, "INPRO_iSS: A component for just-in-time incremental speech synthesis," in *Proceedings of the ACL 2012 System Demonstrations*. Jeju Island, Korea: Association for Computational Linguistics, Jul. 2012, pp. 103–108. [Online]. Available: https://www.aclweb.org/anthology/P12-3018

[3] M. Pouget, T. Hueber, G. Bailly, and T. Baumann, "HMM training strategy for incremental speech synthesis," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[4] 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 *ICASSP*. IEEE, 2018, pp. 4779–4783.

[5] J. Sotelo, S. Mehri, K. Kumar, J. F. Santos, K. Kastner, A. Courville, and Y. Bengio, "Char2wav: End-to-end speech synthesis," *ICLR - Workshop Track*, 2017.

[6] S. Vasquez and M. Lewis, "Melnet: A generative model for audio in the frequency domain," *arXiv preprint arXiv:1906.01083*, 2019.

[7] Y. He, T. N. Sainath, R. Prabhavalkar, I. McGraw, R. Alvarez, D. Zhao, D. Rybach, A. Kannan, Y. Wu, R. Pang *et al.*, "Streaming end-to-end speech recognition for mobile devices," in *ICASSP*. IEEE, 2019, pp. 6381–6385.

[8] Q. Zhang, H. Lu, H. Sak, A. Tripathi, E. McDermott, S. Koo, and S. Kumar, "Transformer transducer: A streamable speech recognition model with transformer encoders and RNN-T loss," *arXiv preprint arXiv:2002.02562*, 2020.

[9] H. Sak, M. Shannon, K. Rao, and F. Beaufays, "Recurrent neural aligner: An encoder-decoder neural network model for sequence to sequence mapping." in *INTERSPEECH*, 2017, pp. 1298–1302.

[10] N. Jaitly, D. Sussillo, Q. V. Le, O. Vinyals, I. Sutskever, and S. Bengio, "A neural transducer," *arXiv preprint arXiv:1511.04868*, 2015.

[11] K. Cho and M. Esipova, "Can neural machine translation do simultaneous translation?" *arXiv preprint arXiv:1606.02012*, 2016. [Online]. Available: http://arxiv.org/abs/1606.02012

[12] J. Gu, G. Neubig, K. Cho, and V. O. Li, "Learning to translate in real-time with neural machine translation," *arXiv preprint arXiv:1610.00388*, 2016.

[13] B. Zheng, R. Zheng, M. Ma, and L. Huang, "Simpler and faster learning of adaptive policies for simultaneous translation," *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, pp. 1349–1354, 2019.

[14] N. Arivazhagan, C. Cherry, W. Macherey, C.-C. Chiu, S. Yavuz, R. Pang, W. Li, and C. Raffel, "Monotonic infinite lookback attention for simultaneous machine translation," in *ACL*, Jul. 2019, pp. 1313–1323.

[15] N. Kalchbrenner, E. Elsen, K. Simonyan, S. Noury, N. Casagrande, E. Lockhart, F. Stimberg, A. van den Oord, S. Dieleman, and K. Kavukcuoglu, "Efficient neural audio synthesis," *PMLR*, vol. 80, pp. 2410–2419, 2018.

[16] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[17] M. Ma, B. Zheng, K. Liu, R. Zheng, H. Liu, K. Peng, K. Church, and L. Huang, "Incremental text-to-speech synthesis with prefix-to-prefix framework," *arXiv preprint arXiv:1911.02750*, 2019.

[18] M. Ma, L. Huang, H. Xiong, K. Liu, C. Zhang, Z. He, H. Liu, X. Li, and H. Wang, "STACL: Simultaneous translation with integrated anticipation and controllable latency," *arXiv preprint arXiv:1810.08398*, 2018.

[19] A. Graves, "Sequence transduction with recurrent neural networks," *arXiv preprint arXiv:1211.3711*, 2012.

[20] C. Chiu and C. Raffel, "Monotonic chunkwise attention," *ICLR*, 2018.

[21] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition," in *ICASSP*. IEEE, 2016, pp. 4945–4949.

[22] A. Tjandra, S. Sakti, and S. Nakamura, "Local monotonic attention mechanism for end-to-end speech and language processing," in *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Taipei, Taiwan: Asian Federation of Natural Language Processing, Nov. 2017, pp. 431–440. [Online]. Available: https://www.aclweb.org/anthology/I17-1044

[23] M. He, Y. Deng, and L. He, "Robust sequence-to-sequence acoustic modeling with stepwise monotonic attention for neural TTS," in *INTERSPEECH*, 2019, pp. 1293–1297.

[24] C. Raffel, M.-T. Luong, P. J. Liu, R. J. Weiss, and D. Eck, "Online and linear-time attention by enforcing monotonic alignments," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 2837–2846.

[25] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International conference on machine learning*, 2015, pp. 2048–2057.

[26] W. Zaremba and I. Sutskever, "Reinforcement learning neural turing machines," *arXiv preprint arXiv:1505.00521*, 2015.

[27] J. Ling, "Coarse-to-fine attention models for document summarization," Ph.D. dissertation, 2017.

[28] 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," 2019.

[29] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[30] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[31] R. S. Sutton *et al.*, *Introduction to reinforcement learning*, 2nd ed., 1998.

[32] D. Elbaz and M. Zibulevsky, "Perceptual audio loss function for deep learning," *arXiv preprint arXiv:1708.05987*, 2017.

[33] A. Mnih and K. Gregor, "Neural variational inference and learning in belief networks," *ICML*, 2014.

[34] D. Silver, "Lectures on reinforcement learning," 2015.

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

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

[37] R. C. Streijl, S. Winkler, and D. S. Hands, "Mean opinion score (MOS) revisited: methods and applications, limitations and alternatives," *Multimedia Systems*, vol. 22, no. 2, pp. 213–227, 2016.

[38] B. Series, *Method for the subjective assessment of intermediate quality level of audio systems*. Geneva: International Telecommunication Union, 2015.

[39] A. Ali and S. Renals, "Word error rate estimation for speech recognition: e-WER," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2018, pp. 20–24.

[40] B. Tranel, "French liaison and elision revisited: A unified account within optimality theory," University of California, Irvine, 1996.

[41] J. Yamagishi, P.-E. Honnet, P. Garner, and A. Lazaridis, "The SIWIS French speech synthesis database," https://datashare.is.ed.ac.uk/handle/10283/2353, 2017.