Improving the Prosody of RNN-based English Text-To-Speech Synthesis by Incorporating a BERT model

Tom Kenter, Manish Sharma, Rob Clark

Google UK

{tomkenter, skmanish, rajclark}@google.com

Abstract

The prosody of currently available speech synthesis systems can be unnatural due to the systems only having access to the text, possibly enriched by linguistic information such as part-of-speech tags and parse trees. We show that incorporating a BERT model in an RNN-based speech synthesis model — where the BERT model is pretrained on large amounts of unlabeled data, and fine-tuned to the speech domain — improves prosody. Additionally, we propose a way of handling arbitrarily long sequences with BERT. Our findings indicate that small BERT models work better than big ones, and that fine-tuning the BERT part of the model is pivotal for getting good results.

1. Introduction

The voice quality of current text-to-speech (TTS) synthesis is getting close to the quality of natural human speech. In terms of prosody, however, the speech that TTS systems produce can still be unnatural along a number of different dimensions including intonation which conveys semantics and emotion. One of the reasons for this is that the input to speech synthesis systems typically consists of only text, read as characters or words [1, 2]. The input is usually enriched by information about syntax, using taggers and parsers [3, 4, 5]. As such, no information about semantics or world knowledge is available to these models, as this is not inferable from just the text. Furthermore, when input is enriched by syntax information, errors made by the algorithms providing this information can propagate through the system, which may degrade, rather than improve, synthesis.

We present CHiVE-BERT, an extension of the Clockwork Hierarchical Variational autoEncoder (CHiVE) prosody model that incorporates a Bidirectional Encoder Representations from Transformers (BERT) network [6], trained first on text, and then fine-tuned on a speech synthesis task – a prosody generation task in particular. The motivation for incorporating a BERT model in our overall model is twofold: i) BERT provides representations that have been proven to embody syntactic information in a more robust way than traditional parsing and tagging [7, 8, 9]; ii) BERT models can provide useful cues beyond syntax, such as word semantics and world knowledge [10].

It has been shown that BERT models can incorporate syntactic information in text-only domain [7, 8]. While the BERT model only implicitly captures the linguistic structure of the input text, we show that its output representations can be used to replace features encoding this syntactic information explicitly.

The BERT models in the CHiVE-BERT system are pretrained on a language modeling task, which is known to lead to semantic information being incorporated into the model [7, 6] We find this enables CHiVE-BERT to learn to correctly pronounce longer noun compounds, such as 'diet cat food', as the knowledge that this is more likely to be interpreted as '(diet (cat food))' rather than '((diet cat) food)', is incorporated into the pretrained model, whereas it is typically hard for standard parsing methods to resolve this ambiguity. Sentences that are challenging linguistically (e.g., because they are long or otherwise difficult to parse) can benefit from the new approach based on BERT representations rather than explicit features, as any errors in the parse information no longer impair performance.

We show that human raters favour CHiVE-BERT over a current state-of-the-art model on three datasets designed to highlight different prosodic phenomena. Additionally, we propose a way of handling long sequences with BERT (which has fixed-length inputs and outputs), so it can deal with arbitrarily long input.

2. Related work

Using BERT in TTS There are existing attempts to use BERT in TTS. Both [11] and [12] employ pretrained BERT models with a Tacotron 2 type TTS models, and show gains in mean opinion scores. [13], again, use pretrained BERT, but only in the front-end text processing components of TTS. They show improved accuracy in the front-end but do not evaluate the final speech quality. In [11, 13], large BERT models are used: 12-layers, and 768 hidden units in each layer (in [12] the size of the BERT models are not specified). In contrast, our proposed model uses a smaller model (only 2 layers, hidden state size 256). More importantly, instead of using the pretrained BERT model as is, we propose to update parameters of the BERT network itself while training our prosody model.

Handling arbitrarily long input with Transformers Perhaps the simplest way of dealing with arbitrarily long input in a Transformer model [14] – which has a fixed input length – is to split the input into consecutive segments, and process these segments separately [15]. An alternative approach is employed in [16, 17], where the input, again, is split into consecutive segments, that the model now recurses over. During evaluation, the Transformer model thus has an input size twice as large as it would usually have.

Different to the approaches described above, but similar to, e.g., [18] we propose to split the input into overlapping segments. This is different from local attention [19, 20, 21] as the Transformer architecture itself is unaltered in our case. For efficiency reasons, we do not recurse over the multiple Transformer outputs [18]. We aggregate the output of overlapping windows by selecting the representations that had most context (i.e. furthest away from the start or end of the window, cf. §3.1.2). This comes at the cost of losing the ability to model prosodic effects spanning very long contexts. We motivate this choice by noting that i) though prosodic phenomena spanning very long contexts can occur, we expect local context to be primarily important for predicting prosodic features; ii) we expect the length of most input sequences to be under the maximum input size of the BERT model, both during fine-tuning and final inference.

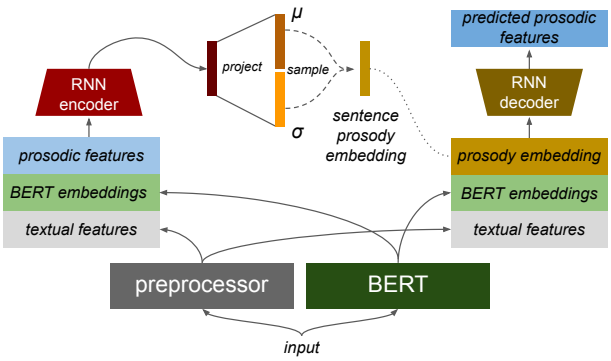

Figure 1: *Schematic overview of the CHiVE-BERT conditional variational auto-encoder model at training time.*

3. CHiVE-BERT

CHiVE-BERT extends the Clockwork Hierarchical Variational autoEncoder (CHiVE) model described in [3]. The CHiVE model is a conditional variational autoencoder (VAE) [22] with inputs and outputs that are prosodic features, namely pitch (fundamental frequency F_0), energy (represented by the 0th cepstral coefficient c_0) and phone durations (in number of frames). The variational embedding represents the prosody of a single utterance. Both the encoder and decoder are structured as hierarchical recurrent neural networks (RNNs) where each layer is clocked dynamically to the linguistic structure of a given utterance. For example, if a particular syllable consists of three phonemes, the encoder phoneme layer will consume three phonemes before its output state is fed to the syllable layer, whereas if the next syllable contains five phonemes, five phonemes will be consumed. Both the encoder and decoder are conditioned on features encoding information about, e.g., syllable stress and part-of-speech (cf. §4.3 for more information on these features). These features are fed to the model at the appropriate linguistic layer by being appended to the state received from the previous layer.
In CHiVE-BERT, the word-level features are replaced by WordPiece embeddings from a pretrained BERT model. This pretrained BERT model is fine-tuned as CHiVE-BERT is trained. Figure 1 shows a high-level overview of CHiVE-BERT. The preprocessor is a text-normalization front-end [23, 24], and is not updated as part of the CHiVE-BERT training procedure. As such, we will not elaborate on it further.

Figure 2 shows the details of the CHiVE-BERT encoder. At the top (shown in red in the figure) is a syllable-rate RNN which encodes an utterance. For each syllable in the utterance it encodes 1) the prosodic features encoded frame by frame (by the blue frame-level RNN in the lower left of the figure), 2) the phone-level features encoded by the phoneme level RNN (in purple, in the lower right), 3) syllable-level features 4) BERT WordPiece embeddings [25], and 5) sentence-level features. The last state of the syllable RNN is fed to the variational layer, which computes a sentence prosody embedding from it.

Similar to the way it works in the encoder, the WordPiece embeddings in the decoder are broadcasted as input to each syllable of a given word, replacing the word-level features in the original CHiVE model. The overall structure of the decoder is somewhat different to that of the encoder to account for the need to predict the number of frames in each phoneme. It is fully described in [3] and not discussed further here for brevity.

3.1. BERT model

The BERT model is run as part of the full model and gradients flow all the way through the model, up until, but not including, the WordPiece embedding lookup table. The input text is split into WordPieces by the default BERT WordPiece tokenizer, and run through the Transformer graph, which outputs an embedding for each WordPiece in the input. As we wish to provide these embeddings to the syllable-level RNNs in the encoder/decoder, which have no notion of WordPieces, the embedding corresponding to the first WordPiece of each word is selected to represent the word, and is presented to each RNN (and replicated for each syllable, cf. Figure 2).

3.1.1. Freezing WordPiece embeddings after pretraining

We freeze the WordPiece embeddings of the model after pretraining [26], as not all WordPieces observed during pretraining might be observed (as often) during the fine-tuning stage. This might cause discrepancies in the way the semantic space would be updated during fine-tuning if the gradients would, in fact, flow through it. If two WordPieces are close after pretraining, but only one of them is observed frequently during fine-tuning, they might end up being separated. This separation, however, would not be informed by the fine-tuning process, but, rather, by data imbalance.

3.1.2. Handling arbitrarily long input with BERT

While the RNN layers of the CHiVE-BERT model can handle arbitrarily long input (memory permitting) as they unroll dynamically, the BERT model has a fixed length m. We augment the BERT model to handle arbitrarily long sequences of WordPieces by splitting a sequence $wp_1, wp_2, \ldots, wp_n$ into windows of size m, if $n > m$, and presenting these windows to the model as a mini-batch. The windows overlap by stride size s, which we set to be (the floor of) half of the input size: $\lfloor (m-2)/2 \rfloor$. Each window w_i, for $i \in \{0, 1, \ldots, \lceil \frac{n+2-m}{s} \rceil\}$ is defined as:

$$w_i = \{\mathit{START}, wp_{i \cdot s+1}, wp_{i \cdot s+2}, \ldots, wp_{i \cdot s+(m-2)}, \mathit{END}\},$$

where wp_i is the WordPiece at position i, and *START* and *END* are start and end tokens, respectively. We introduce two new tokens `[BREAK]` and `[CONT]`. The *START* token of the first window is the standard `[CLS]` token, and `[CONT]` for any subsequent window. The *END* token of the last window is the standard end-of-sequence token `[SEP]` (which can occur earlier in the window, in which case it is followed by padding to fill up to size m), and a `[BREAK]` token for any window before it.

To combine the $\lceil \frac{n+2-m}{s} \rceil + 1$ overlapping windows of length m to a sequence of length n again, we select the center $\lfloor s/2 \rfloor$ WordPiece embeddings from every window, except for the first one, where we start from index 0, and the last one, where we end at the `[SEP]` token, and pad the remainder.

4. Experimental setup

Here we describe the way our experiments are set up.

4.1. CHiVE-BERT model

The CHiVE-BERT model is trained on 215,650 utterances (corresponding to about 228 hours of speech). A development set of 3000 is employed to tune hyperparameters. The data was curated to have good phonetic coverage, and was recorded under studio conditions by 30 native US English speakers. While

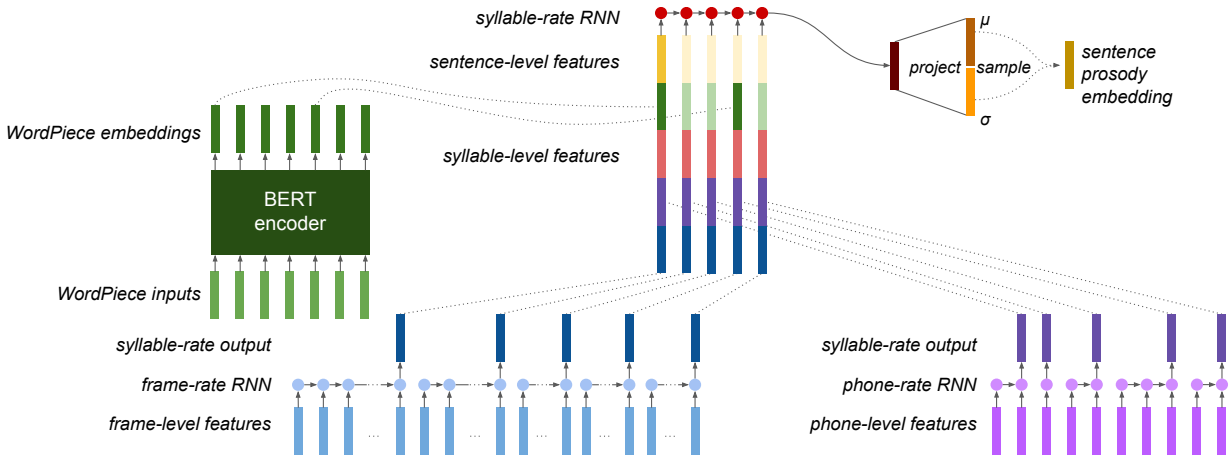

Figure 2: *CHiVE-BERT encoder and variational layer. Circles represent blocks of RNN cells, rectangles represent vectors. Broadcasting (vector duplication) is indicated by vectors being displayed in a lighter shade. The preprocessor, that computes the input features at various levels, is omitted from the picture, as it is run offline and, as such, not part of the network. (Best viewed in colour).*

some speakers are represented better than others, all speakers are represented by a substantial number of utterances. Alignment, down to phoneme level, was done automatically.

The BERT part of the CHiVE-BERT model is trained on the same corpus[1] the publicly available **BERT$_{\text{BASE}}$** model was trained on[2] [6], using a dropout probability of 0.1 [27], Gaussian Error Linear Unit (GELU) activations [28], hidden size 256, intermediate size 1024, 4 attention heads, 2 hidden layers, and a vocabulary size of 30,522. The model has an input size of 512 (m in §3.1.2), i.e., it reads 512 WordPiece embeddings at once.

The internal size of the syllable-level RNN and of the sentence prosody embedding (the output of the variational layer) is 256. The internal size of both the frame-rate RNN and the phone-rate RNN is 64 in the encoder, and 64 and 32, respectively, in the decoder. We represent F_0 and c_0 values at 200Hz (i.e., we use 5ms frames). Adadelta optimization is used [29], with a decay rate $\rho = 0.95$ and $\epsilon = 10^{-6}$, and an initial learning rate of 0.05, exponentially decayed over 750,000 steps. All networks are trained for 1M iterations, with a batch size of 4.

As reported in, e.g., [30], the fine-tuning process is sensitive to different initializations. We run 10 runs for each hyperparameter setting considered, and report on the model performing best on the development set.

4.2. Baseline

The baseline model is the prosody model currently used for popular US English voices in the Google Assistant, described in [3]. The capacity of this model (the number of parameters) is lower than the capacity of the CHiVE-BERT model, as the latter incorporates an extra Transformer network. It might be considered fair, therefore, to compare the CHiVE-BERT model to a version of the baseline that is similar in terms of capacity. We found, however, in preliminary experiments, that increasing the size of the hidden states, or number of RNN cells in any of the layers decreased performance, as the models started to overfit. As such, the architecture of the baseline is tuned to yield the highest performance in terms of MOS score, and hence is a strong baseline to compare the CHiVE-BERT model against.

[1] See `https://github.com/google-research/bert`.

[2] We also tried initializing from **BERT$_{\text{BASE}}$** itself in preliminary experiments. However, we found that did not yield promising results, and as it takes a long time to fine-tune, we left it out of our final experiments.

4.3. Features

Both CHiVE-BERT and the baseline model have input features describing the textual contents at the level of sentence, words (only for the baseline), syllables and phonemes, following, e.g., [5]. The sentence-level features contain information about the speaker and their gender. Word-level features contain information about POS tags, dependency parse, phrase-level features (indicating if the phrase is part of a question, statement), and preceding or subsequent punctuation marks. At syllable level, information such as number of phonemes and lexical stress is represented. Lastly, the phoneme-level features contain information about the position of the phoneme in the syllable, the phoneme identity, and the number of phonemes in the syllable.

CHiVE-BERT has the same input features the baseline model has, except for the features at word level, which are left out and replaced by BERT representations as described in §3.[3]

4.4. WaveNet

We use two WaveNets [31]: one for our baseline, and one for the CHiVE-BERT model. Both are trained on linguistic and prosodic features, using a dataset containing 115,754 utterances of speech for 9 speakers. They are trained on multiple speakers because we found that even if the models are used to generate speech for only one or two speakers as they are in our experiments, training them on more speakers yields better results. The only difference between the two WaveNets is that the one used for the baseline does have access to features at word level, while the one used for CHiVE-BERT does not, to prevent it from getting potentially conflicting information between the BERT representation and the (noisy) word-level linguistic features.

4.5. Evaluation

To evaluate the performance of CHiVE-BERT compared to the baseline model we run 7-way side-by-side tests, where raters indicate which audio sample they think sounds better on a scale from -3 to +3, where the center of the scale means "no difference". Every rater rates at most 6 samples, and every sample is

[3] In preliminary experiments, we also tried combining both word-level linguistic features and BERT representations, but as the results were not promising, we left out this setting in the final experiments.

Table 1: *Results of comparing the CHiVE-BERT model to the baseline. All results are statistically significant, using a two-sided t-test with $\alpha = 0.01$. The ♀ and ♂ symbols indicate female and male speaker, respectively.*

		test outcome
Hard lines	♀	0.549 ± 0.076
	♂	0.524 ± 0.077
Questions	♀	0.287 ± 0.077
	♂	0.229 ± 0.075
Generic lines	♀	0.274 ± 0.058
	♂	0.186 ± 0.056

Table 2: *Mean absolute F_0 error in Hz, for CHiVE-BERT models incorporating BERT models of different sizes.*

hidden state	interm. state	# attention heads	# Transformer layers	F_0 error
24	96	1	2	20.7
128	512	2	2	18.11
128	512	2	4	17.77
256	**1024**	**4**	**2**	**17.59**
256	1024	4	4	22.28
768	3072	12	6	22.51
768	3072	12	8	22.66

rated by 1 to 8 raters. To gain insight into the quality on different kinds of input data, we test it on three separate datasets:

Hard lines A test set of 286 lines, expected to be hard, containing, e.g., titles and long noun compounds;
Questions As questions are prosodically markedly different from statements, we include a test sets of 299 questions;
Generic lines The aim of this set is to test against regression (i.e., that normal lines that already sound good are not negatively impacted). The set consists of 911 lines, typically short, for which a generic default prosody is likely to be appropriate.

5. Results and analysis

As described in §4.5 we perform tests on three datasets. The results of the tests are in Table 1. The difference between CHiVE-BERT and the baseline is most apparent for the hard lines, and is still significant for the other test sets.
Audio samples of the CHiVE-BERT model and the baseline are available at `https://google.github.io/chive-prosody/chive-bert/`.

5.1. Model sizes

When finetuning for specific NLP task, bigger BERT (or BERT-like) models tend to yield better results than smaller models [32, 33]. This was not the case in our experiments, similar to findings in, e.g., [34]. Table 2 lists the mean absolute error, in Hz, of predicted F_0 when the decoder is conditioned on an all-zeros embedding, for CHiVE-BERT models incorporating BERT models of different sizes (pretrained on the same corpus, and all fine-tuned). We observe a trend where bigger models yield higher losses. To see if smaller is always better we also tried a very small model (first row in the table) which does surprisingly well, but worse than the medium sized models.

Table 3: *Comparison of CHiVE-BERT to the same model in which the BERT part is not fine-tuned. All results are statistically significant, using a two-sided t-test with $\alpha = 0.01$*

		test outcome
Hard lines	♀	0.320 ± 0.069
	♂	0.642 ± 0.072
Questions	♀	0.238 ± 0.082
	♂	0.504 ± 0.083
Generic lines	♀	0.208 ± 0.054
	♂	0.588 ± 0.068

5.2. Fine-tuning

Different from the findings in [13], but in line with, e.g., [35], taking the output of a fixed BERT model did not improve performance in our setting. We found that the BERT part of the model has to be fine-tuned. Table 3 lists the results of a side-by-side test of the fine-tuned model described above, and the exact same model where no gradients flow to the BERT part of the model at all. As we can see from this table, fine-tuning the model consistently yields better results.

5.3. Negative findings

BERT models typically have a much smaller learning rate during pretraining (e.g., 10^{-4}) than the one we use when training CHiVE-BERT (0.05). We tried applying a similar (smaller) learning rate to just the BERT part of the model, which yielded identical or worse results than not doing this.

6. Conclusion

We presented CHiVE-BERT, an RNN-based prosody model that incorporates a BERT model, fine-tuned during training, to improve the prosody of synthesized speech. We showed that our model outperforms the state-of-the-art baseline on three datasets selected to be prosodically different, for a female and male speaker.

We conclude from our findings that, rather than using embeddings from a pretrained BERT model, fine-tuning BERT is crucial for getting good results. Additionally, our experiments indicate that relatively small BERT models yield better results than much bigger ones, when employed in a speech synthesis context as proposed. This is at odds with findings in other NLP tasks, where bigger models tend to yield better performance.

A limitation of the work presented is that, even though both the RNNs and the BERT model we implemented can, in theory, handle arbitrarily long input [36], very long sequences will take prohibitively long to synthesize. Novel developments in model distillation, or new ways of running parts of the network in parallel might allow for longer sequences to be dealt with.

Lastly, the current evaluations were done on US English data only. It would be interesting to see if similar improvements can be obtained on other languages.

7. Acknowledgments

The authors would like to thank the Google TTS Research team, in particular Vincent Wan and Chun-an Chan, for their invaluable help at every stage of the development of the work presented here, Aliaksei Severyn for helping out with all things BERT-related, and Alexander Gutkin for helpful comments on earlier versions of the manuscript. Lastly, many thanks to the reviewers for their thoughtful and constructive suggestions.

8. References

[1] 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 *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 4779–4783.

[2] 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," *Interspeech 2017*, pp. 4006–4010, 2017.

[3] V. Wan, C. an Chan, T. Kenter, J. Vit, and R. Clark, "CHiVE: Varying prosody in speech synthesis with a linguistically driven dynamic hierarchical conditional variational network," in *International Conference on Machine Learning (ICML)*, 2019, pp. 3331–3340.

[4] H. Zen, Y. Agiomyrgiannakis, N. Egberts, F. Henderson, and P. Szczepaniak, "Fast, compact, and high quality LSTM-RNN based statistical parametric speech synthesizers for mobile devices," in *Interspeech 2016*, 2016, pp. 2273–2277.

[5] H. Zen and H. Sak, "Unidirectional long short-term memory recurrent neural network with recurrent output layer for low-latency speech synthesis," in *Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 4470–4474.

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019, pp. 4171–4186.

[7] G. Jawahar, B. Sagot, and D. Seddah, "What does BERT learn about the structure of language?" in *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019, pp. 3651–3657.

[8] I. Tenney, D. Das, and E. Pavlick, "BERT rediscovers the classical NLP pipeline," in *Meeting of the Association for Computational Linguistics (ACL)*, 2019, pp. 4593–4601.

[9] J. Hewitt and C. D. Manning, "A structural probe for finding syntax in word representations," in *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019, pp. 4129–4138.

[10] F. Petroni, T. Rocktäschel, S. Riedel, P. Lewis, A. Bakhtin, Y. Wu, and A. Miller, "Language models as knowledge bases?" in *EMNLP-IJCNLP*, 2019, pp. 2463–2473.

[11] T. Hayashi, S. Watanabe, T. Toda, K. Takeda, S. Toshniwal, and K. Livescu, "Pre-trained text embeddings for enhanced text-to-speech synthesis," *Interspeech 2019*, pp. 4430–4434, 2019.

[12] Y. Xiao, L. He, H. Ming, and F. K. Soong, "Improving prosody with linguistic and BERT derived features in multi-speaker based Mandarin Chinese neural TTS," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 6704–6708.

[13] B. Yang, J. Zhong, and S. Liu, "Pre-trained text representations for improving front-end text processing in Mandarin text-to-speech synthesis," *Interspeech 2019*, pp. 4480–4484, 2019.

[14] 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 Proceeding Systems*, 2017, pp. 5998–6008.

[15] R. Al-Rfou, D. Choe, N. Constant, M. Guo, and L. Jones, "Character-level language modeling with deeper self-attention," in *AAAI Conference on Artificial Intelligence*, 2019, pp. 3159–3166.

[16] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. Le, and R. Salakhutdinov, "Transformer-XL: Attentive language models beyond a fixed-length context," in *57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 2978–2988.

[17] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "XLNet: Generalized autoregressive pretraining for language understanding," in *Advances in Neural Information Processing Systems*, 2019, pp. 5754–5764.

[18] R. Pappagari, P. Zelasko, J. Villalba, Y. Carmiel, and N. Dehak, "Hierarchical transformers for long document classification," in *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2019, pp. 838–844.

[19] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," *arXiv preprint arXiv:2004.05150*, 2020.

[20] M. Zaheer, G. Guruganesh, A. Dubey, J. Ainslie, C. Alberti, S. Ontanon, P. Pham, A. Ravula, Q. Wang, L. Yang *et al.*, "Big bird: Transformers for longer sequences," *arXiv preprint arXiv:2007.14062*, 2020.

[21] S. Hofstätter, H. Zamani, B. Mitra, N. Craswell, and A. Hanbury, "Local self-attention over long text for efficient document retrieval," in *International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2020)*, 2020.

[22] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *International Conference on Learning Representations (ICLR)*, 2014.

[23] P. Ebden and R. Sproat, "The Kestrel TTS text normalization system," *Natural Language Engineering*, vol. 21, no. 3, pp. 333–353, 2015.

[24] H. Zhang, R. Sproat, A. H. Ng, F. Stahlberg, X. Peng, K. Gorman, and B. Roark, "Neural models of text normalization for speech applications," *Computational Linguistics*, vol. 45, no. 2, pp. 293–337, 2019.

[25] M. Schuster and K. Nakajima, "Japanese and Korean voice search," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 5149–5152.

[26] S. Wu and M. Dredze, "Beto, Bentz, Becas: The surprising cross-lingual effectiveness of BERT," in *EMNLP-IJCNLP*, 2019, pp. 833–844.

[27] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, pp. 1929–1958, 2014.

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

[29] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[30] J. Dodge, G. Ilharco, R. Schwartz, A. Farhadi, H. Hajishirzi, and N. A. Smith, "Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping," *ArXiv*, vol. abs/2002.06305, 2020.

[31] A. v. d. Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. v. d. Driessche, E. Lockhart, L. C. Cobo, F. Stimberg *et al.*, "Parallel WaveNet: Fast high-fidelity speech synthesis," in *International Conference on Machine Learning*, 2017, pp. 3918–3926.

[32] N. F. Liu, M. Gardner, Y. Belinkov, M. E. Peters, and N. A. Smith, "Linguistic knowledge and transferability of contextual representations," in *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019, pp. 1073–1094.

[33] A. Rogers, O. Kovaleva, and A. Rumshisky, "A primer in BERTology: What we know about how BERT works," *arXiv preprint arXiv:2002.12327*, 2020.

[34] Y. Goldberg, "Assessing BERT's syntactic abilities," *arXiv preprint arXiv:1901.05287*, 2019.

[35] M. E. Peters, S. Ruder, and N. A. Smith, "To tune or not to tune? adapting pretrained representations to diverse tasks," in *Workshop on Representation Learning for NLP (RepL4NLP-2019)*, 2019, pp. 7–14.

[36] R. Clark, H. Silen, T. Kenter, and R. Leith, "Evaluating long-form text-to-speech: Comparing the ratings of sentences and paragraphs," in *ISCA Speech Synthesis Workshop (SSW10)*, 2019.