BREATHING AND SPEECH PLANNING IN SPONTANEOUS SPEECH SYNTHESIS

Éva Székely, Gustav Eje Henter, Jonas Beskow, Joakim Gustafson

Division of Speech, Music and Hearing, KTH Royal Institute of Technology, Stockholm, Sweden

ABSTRACT

Breathing and speech planning in spontaneous speech are coordinated processes, often exhibiting disfluent patterns. While synthetic speech is not subject to respiratory needs, integrating breath into synthesis has advantages for naturalness and recall. At the same time, a synthetic voice reproducing disfluent breathing patterns learned from the data can be problematic. To address this, we first propose training stochastic TTS on a corpus of overlapping breath-group bigrams, to take context into account. Next, we introduce an unsupervised automatic annotation of likely-disfluent breath events, through a product-of-experts model that combines the output of two breath-event predictors, each using complementary information and operating in opposite directions. This annotation enables creating an automatically-breathing spontaneous speech synthesiser with a more fluent breathing style. A subjective evaluation on two spoken genres (impromptu and rehearsed) found the proposed system to be preferred over the baseline approach treating all breath events the same.

Index Terms— Speech synthesis, spontaneous speech, breathing, speech planning, ensemble method

1. INTRODUCTION

Humans necessarily breathe when we speak, and integrating breathing into speech synthesis has long been considered desirable, cf. [1, 2]. Breathing not only adds to the perceived naturalness of synthetic speech over using silent pauses [3], but also breaks the message into smaller chunks that may be easier to comprehend [4] and improves recall [5]. Moreover, TTS systems that integrate breathing into the synthesis and allow it to be controlled have the potential to make virtual characters [6] and social robots more convincing.

In natural spontaneous speech, breathing and speech planning operate in a system of coordinated processes. People tailor the inhalation location and depth to their linguistic plan, but the linguistic plan itself is also affected by the speaker's respiratory state [7]. While breath events in read speech are largely organised around phrase and sentence boundaries [8], comparison studies between read and spontaneous speech have shown that the latter exhibits a higher percentage of breath events placed within phrases, i.e., in ungrammatical locations [9]. Moreover, the speaking style, communicative situation and the level of cognitive demand have been shown to affect breathing [10, 11, 12]. [13] differentiates between alternating periods of fluency and periods of hesitation; when speech production is disfluent, breathing patterns also reflect that ongoing breakdown of linguistic planning.

This research was supported by Swedish Research Council project Connected: context-aware speech synthesis for conversational AI, VR (2019-05003), the Swedish Foundation for Strategic Research project EACare (RIT15-0107), and the Wallenberg AI, Autonomous Systems and Software Program (WASP).

Synthetic speech is, evidently, not constrained by physiological needs like respiratory intake, nor is it dependent on cognitive demands related to the availability of the linguistic plan, since the full text to be spoken is typically provided up front. We can therefore propose that, from the speech synthesis point of view, some inhalation breaths can be considered disfluencies, and exist in the data due to breakdowns of the coordinated planning and breathing processes. Selectively choosing not to reproduce such breath events in synthesis would lead to more fluent and appealing output speech.

1.1. Contributions and relation to prior work

In this paper we study breathing in spontaneous speech synthesis, and how one can create a breathing speech synthesiser where the number of idiosyncratic breath events – pervasive in spontaneous speech corpora – is reduced. Our approach to this end includes several contributions: First, to enable TTS to produce longer speech segments that also better evidence the impact of synthetic breathing strategies, we propose training attention-based neural TTS on pairs (bigrams) of consecutive breath groups (Sec. 2.1). Second, we furthermore propose to use the output of two deep-learning-based breath predictors, fused using an ensemble approach (Sec. 2.4), to automatically annotate breath events in the bigram database according to their probable function. While the sub-classifiers are trained using supervised learning, the method itself is unsupervised in that it requires no additional annotation compared to what is needed to create the TTS in the first place. In particular, there is no manual labelling of breath events into different classes. Using the resulting breath-type annotation in training a stochastic speech synthesiser yields automatically-breathing TTS where specific types of breath events can be excluded from the synthesis.

Aside from the fact that interactions between breathing and speech planning have not been studied in speech synthesis before, our method differs from prior work on breath prediction and synthesis in several other ways: Unlike [3, 14] (but similar to [1, 2, 15]), we specifically consider spontaneous speech. We also distinguish between several types of breath events, as seen only in [3]. Moreover, our attention-based synthesiser learns to insert breath events in a joint end-to-end manner and does not use a separately trained, text-level "language model" of breath locations. [1, 2, 3, 14]. This joint learning to speak and breathe produces a synthesiser where prosody, voice quality, and breath are coordinated, for instance sounding like the speaker speeding up and using expiratory reserve volume in long stretches of speech without a breath. (For audio, see our evaluation samples linked in Sec. 4.1.)

2. PROPOSED METHOD

This section describes our approach for creating breathing, spontaneous TTS that speaks connected utterances in a more fluent manner than the training data itself. We first explain (in Sec. 2.1) how we

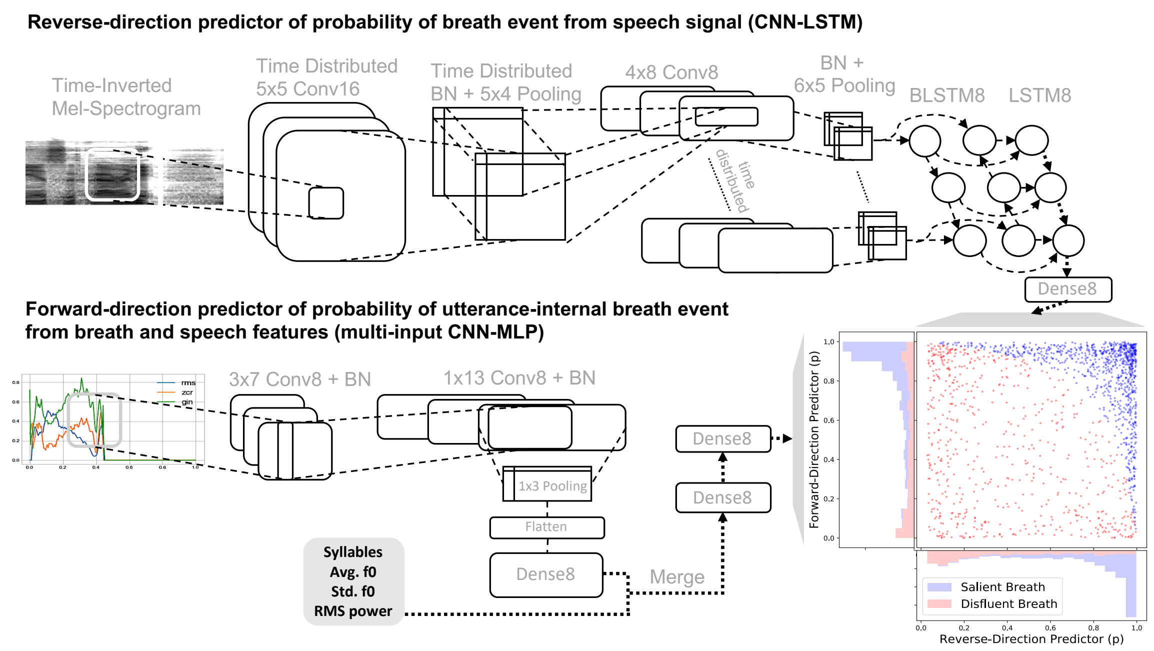
Reverse-direction predictor of probability of breath event from speech signal (CNN-LSTM)
Time-Inverted Mel-Spectrogram
Time Distributed 5x5 Conv16
Time Distributed BN + 5x4 Pooling
4x8 Conv8
BN + 6x5 Pooling
BLSTM8
LSTM8
time distributed
Dense8
Forward-direction predictor of probability of utterance-internal breath event from breath and speech features (multi-input CNN-MLP)
rms
zcr
gin
3x7 Conv8 + BN
1x13 Conv8 + BN
1x3 Pooling
Flatten
Dense8
Syllables
Avg. f0
Std. f0
RMS power
Merge
Dense8
Dense8
Forward-Direction Predictor (p)
Reverse-Direction Predictor (p)
Salient Breath
Disfluent Breath

Fig. 1. Architecture of the two classifiers, including selected layer sizes, and a scatter plot of breath-event probability estimates for the corpus.

augment the training data to allow contextual information to flow between individual utterances in the corpus. After that we describe two breath-event predictors (one forwards, in Sec. 2.2, and one backwards in time, in Sec. 2.3). In Sec. 2.4 we then combine these detectors using an ensemble-based method to create an unsupervised method for separating salient from disfluent breath events in the augmented data. Essentially, by identifying breaths whose locations are difficult for a machine to predict from speech acoustics, and then not reproducing these, likely disfluent, breath events during synthesis, we hope to eliminate random and idiosyncratic machine breathing from the output speech and thus obtain a more fluent and appealing stochastic speech synthesiser.

2.1. Bigram corpus method

Despite the interest in TTS that speaks longer chunks of text, e.g., [16], no major engine currently assumes a connection between consecutive audio files in the corpus. Instead, utterances are traditionally treated in isolation. The main reasons for retaining this custom is that 1) most conventional TTS training material is comprised of sentences read in isolation, and 2) loading longer speech segments (e.g., audiobook paragraphs instead of sentences) into a conventional TTS engine would quickly result in the system running out of memory. This state of affairs is unfortunate in the case of corpora created from longer continuous recordings of speech, such as audiobooks [16, 17]. It is even more problematic for spontaneous speech, where syntactic forms do not follow the conventions of written language and cannot be easily separated into standalone, coherent semantic units. Therefore, whatever criteria the segmentation is based on, the resulting segments are to be interpreted in context.

We here propose a simple data-augmentation method that allows for context information to be transmitted across utterances during the training process, while still limiting segment length and thus avoiding the synthesiser running out of memory. Concretely, we propose to pair the individual breath groups in the corpus into a sequence of overlapping, utterance-level bigrams. If $\{(b_n, u_n, b_{n+1})\}_{n=1}^{N}$ is a training set of utterances u delineated by breath events b, we propose to train on $\{(b_n, u_n, b_{n+1}, u_{n+1}, b_{n+2})\}_{n=1}^{N-1}$, i.e., all concatenations of two contiguous breath groups. We will refer to this as the *bigram-corpus method* and the bigrams themselves as *double breath groups*, since they start and end with a breath event, with an additional breath event somewhere in the middle. In practice, our approach means that we duplicate training data, creating longer input utterances that allow the synthesiser to see each utterance in context twice: once together with the preceding and once together with the following utterance. Training a breath-group-based spontaneous speech synthesiser with the bigram-based data augmentation technique enables the synthesiser to: a) speak longer utterances, b) place breath events within an utterance, and c) take context into account.

2.2. Forward-direction predictor of utterance-internal breaths

The first classifier predicts whether the initial breath and summary characteristics of the subsequent speech are most consistent with a single breath group or with two consecutive breath groups. We then use the predictions of the trained classifier as an indication of which double breath groups in our corpus could likely have been completed with their initial inhalation alone. This lets us annotate breath events as more likely to be unnecessary, and thus candidates for exclusion in order to exhibit a more fluent and rehearsed speaking style.

To characterise the initial breath event, three feature vectors are used as input for two convolutional layers with batch-normalisation:

- root-mean-square (RMS) power of the breath signal, using a window of 20 ms and a step size of 5 ms;
- zero-crossing rate (ZCR), the number of times the audio waveform changes sign divided by the total number of samples in the window (windowing same as above); and
- gradient index (GI), a parameter based on the gradient of the signal at each change of direction, which is used to characterise breath and to differentiate on level of voicing [18].

Duration of the breath was modelled implicitly by the length of the input vectors. The output of the CNN layers was flattened after max-pooling and taken through a fully connected layer. The output was merged with the following summary statistics of the speech: the number of syllables, the mean and standard deviation of f_0, and the average RMS power of the speech signal, windowed like the breath features. Two fully connected layers then mapped the merged vector to the softmax output layer of the model. This architecture is visualised in the bottom row of Fig. 1.

2.3. Reverse-direction predictor of breath events from speech

The second classifier uses acoustic information to estimate the probability that a speech segment was preceded by a breath. This is used to assess the acoustic-prosodic salience of breath events given the subsequent speech, hence the classifier moves from right to left.

Inputs to the reverse-direction predictor are mel-scaled spectrograms of two-second long speech signals, inverted on the time axis, as the breath for the positive examples precedes the sample (with 0.05 second lag to ensure no breath is accidentally captured in the speech sample). Negative examples for training are taken from a disjoint group of speech segments, equally many as the positive examples, at least one second after a breath. The predictor architecture (illustrated in the top row of Fig. 1) is similar to that of the breath detector presented in [19], but uses a one-directional LSTM, as the presence of a breath is tested for at the beginning of the sample. The spectrograms are also sliced into longer, 0.25 second segments and span of the convolutions is increased to better identify acoustic trends. Two convolutional (CNN) layers with batch-normalisation and max-pooling are followed by a bi-directional (BLSTM) recurrent and a one-directional recurrent (LSTM) layer. A fully connected layer connects the outputs of the LSTM to the softmax output layer of the model. By training a reverse predictor on one part of the corpus, its predictions can be on the other part of the material to annotate the acoustic-prosodic salience of the middle breaths in each double breath group, and vice versa.

2.4. Classifier ensemble

The two classifiers above provide complementary information about whether or not a given breath event might be considered disfluent. This suggests that they can be combined into a classifier ensemble that is likely to produce better decisions than a single classifier on its own. Such ensembles can be built in different ways; we opt to combine the two classifiers using a *product of experts* approach [20], in which the probabilities from the two classifiers are multiplied together for each class and then renormalised. Mathematically, p_e, the probability of a breath according to the ensemble, is computed as

$$p_e = \frac{p_1 p_2}{p_1 p_2 + (1 - p_1)(1 - p_2)}, \qquad (1)$$

where p_1 and p_2 are the breath probabilities returned by the two constituent predictors. Products of experts are central to parameter-generation algorithms in statistical parametric speech synthesis [21] and have been used to identify voice styles in audiobooks [22]. Whereas in an additive mixture of experts the most uncertain classifier has a substantial impact on the combined class probabilities, a product-of-experts model instead means that the most certain classifier exerts the greatest influence over the ensemble output. The scatter plot in the bottom right of Fig. 1 shows the classifications made on our data, and the decision boundary of the ensemble described in Sec. 3.2. Note that an additive mixture of experts would exhibit a linear decision boundary sloping downward 45 degrees.

As described in [23], not annotating acoustic events, such as filled pauses (uh/um), with Tacotron 2 leads to TTS that randomly produces such acoustic events on its own accord in likely locations. Annotating acoustic events, in contrast, allows control over them, and will only insert them into the synthetic speech if explicitly requested [23]. By *only annotating the disfluent breaths* (using a unique symbol added to the phone set) and then not inserting such breaths into synthesis prompts, we obtain a synthesiser that automatically inserts salient breath events at likely locations, without producing disfluent breathing.

3. DATABASE AND SYNTHESIS

3.1. Corpus, segmentation, and transcription

For voice training, we used the audio from the Trinity Speech-Gesture Dataset [24], 25 impromptu monologues, on average 10.6 minutes long, performed over multiple recording sessions by a male speaker of Irish English. The actor is speaking in a colloquial style, spontaneously and without interruption, on topics such as hobbies, daily activities, and interests. During the monologues, he addresses a person seated behind the cameras who is giving visual, but no verbal, feedback. The last monologue was held out for evaluation.

The corpus was transcribed using the enhanced video model of Google Cloud Speech-to-Text API [25]. The Gentle forced aligner [26] and the US English BroadbandModel of the IBM Watson Speech-to-Text were used to detect and disambiguate between filled pauses *uh* and *um* in order to obtain an annotation as close as possible to the original speech. For more details on the annotation pipeline please refer to [15]. All punctuation was removed from the automatic transcription, as ASR inserts punctuation with the main goal of making text more readable, which need not match the prosody of the realised speech. A simplified version of the speaker-dependent breath detector proposed by [19] was employed to detect breath events in the corpus, with which we were able to segment the audio recordings fully automatically to produce a TTS corpus of 3,487 breath-group utterances.

3.2. Unsupervised breath-event annotation

We used the product-of-experts methodology described in Sec. 2.4 to annotate likely disfluent breath events in the corpus, by combining the output of the forward (Sec. 2.2) and backward (Sec. 2.3) acoustic-based breath event predictors. Since we wanted to use the corpus to train classifiers, and then use those classifiers in turn as an annotation method for the same corpus, a naïve application risks feeding the system training examples at test time. To prevent this we used a cross-validation approach, where classifiers trained on one half of the database were used to annotate the other, and vice versa. 20% of these training sets were further held out as development sets for the individual classifiers.

All classifier networks were randomly initialised and trained for 40 epochs to minimise cross-entropy using Adadelta with default parameters. For the forward-looking predictor we winsorised all input features (breath and speech) at the upper 99th percentile to remove outliers and then normalised to a range between 0 and 1. The training accuracy of the forward predictor was 0.88 and 0.86 for each half, with development set accuracy at 0.76 and 0.77, respectively. The same numbers for reverse-direction predictor were 0.84 and 0.87 on training data and 0.74 and 0.72 on the dev set.

Of the 3,338 double breath groups identified in the training data, 1,873 had a total speech duration of at most 7.9 seconds (the 95th percentile of single-breath group durations) which made them candidates for assigning a disfluency label to the breath event, since they conceivably could have been completed in one breath. To generate a breath probability to feed into the product-of-experts ensemble also from the the backward-looking classifier (which operates on acoustic windows rather than summary statistics), the classifier was applied to the first two seconds (padded with silence if necessary) following the middle breath of each such double breath group. Taking the average validation accuracy of the 4 models as a heuristic for the number of target breaths to mark as a disfluency, a cut-off point of 0.9 was set for the product-of-experts classifier. This flagged 28% of breath events as likely to be disfluent. If two consecutive breath events fell below the threshold, only the one with a lower probability was chosen, giving the final annotation of 614 breath events as disfluent, which were labelled when training the proposed TTS system.

3.3. Systems built

All systems trained used the bigram utterance structure described in Sec. 2.1 and the versions were given descriptive names according to the labelling of breath events in the training data and in the synthesis prompt.

1. **AutoBreathe Baseline** (ABB): no breath events labelled during training or inserted in the input prompts;
2. **GroundTruthBreathe** (GTB): all breath events labelled, ground-truth breath locations inserted into the prompts;
3. **NoBreathe** (NoB): all breath events labelled, but no breaths inserted into the prompts;
4. **AutoBreathe Proposed** (ABP): only the disfluent breath events labelled, no breaths inserted into the prompts.

Because ABP and ABB are trained on data where some or all breath events do not have a corresponding token in the transcription, they both insert breath events automatically, rendered by the TTS engine according to their distribution learned during training (cf. [23]). We note that NoB still inserts a breath at the beginning and end of each synthesised utterance, but never within.

The voices described here were built using the implementation [27] of the Tacotron 2 spectrogram-prediction framework [28]. Audio was sampled at 22.05 kHz. The Griffin-Lim algorithm [29] was used for waveform synthesis. Transfer learning on a pretrained model of read speech [30] and front-end-based phonetisation were used as these have been shown to reduce the pronunciation errors produced by voices trained on corpora transcribed by ASR [15].

4. PERCEPTUAL EVALUATION

4.1. Evaluation design

We evaluated the naturalness of the different TTS breathing strategies on two styles of spontaneous speech: ten utterances from the final, held-out monologue in our corpus (representing impromptu speech) and ten utterances from an episode in the Project Debater corpus [31] (which are more rehearsed, but still spontaneous). All test utterances were 2–4 breath groups long (6–14 seconds), where the cut-off was chosen based on semantic content. Subjects were given a MUSHRA-like listening test implemented using the WebMUSHRA codebase [32], and asked to rate samples from the different systems speaking the same text prompt naturalness on a scale from 0 (worst) to 100 (best), paying particular attention to breathing. The evaluation samples are available at www.speech.kth.se/tts-demos/icassp20.

4.2. Results

40 native speakers of English recruited through Prolific Academic participated in the test. All participants reported having used headphones. Results are graphed in Fig. 2. Based on pair-wise Wilcoxon signed-rank tests, AutoBreathe Proposed was found to perform significantly better than the AutoBreathe Baseline ($p<0.01$ overall) and NoBreathe ($p\ll 0.01$ overall) in both categories. In the held-out category, no significant difference was observed between GroundTruthBreathe and ABP ($p=0.32$). In the debate category, GTB was significantly preferred over any other system ($p<0.01$).

To verify that better scores cannot be explained simply by how often a system places a breath, we fitted a linear model to the scores, using a system variable (categorical) and the number of syllables per breath group in the sample (numerical) as independent variables. In both the held-out and the debate categories, the system remained a significant variable ($p\ll 0.01$), while the effect of the number of syllables per breath group was not significant ($p\gg 0.05$).

Taken together, the fact that ABB is consistently worse than both GTB and ABP suggests that the baseline approach does not model some breath events in a convincing manner, and that our approach to exclude likely disfluent breath events from the synthesis improves perceived naturalness. Even though ABB and NoB were rated similarly on the debate prompts, it was likely for different reasons, with NoB often audibly running out of breath towards the end, while ABB instead sometimes displaying a suboptimal breath-placement strategy. That ABP was preferred over ABB on the debate utterances suggests that the more fluent, proposed breathing style was a better fit for prompts from a more rehearsed spoken genre, but still not as good as the debaters' own respiratory patterns. This shows that both breathing-control schemes (automatic and manual) can adjust the level of fluency in the generated speech.

5. CONCLUSIONS

This study investigated breathing and speech planning in the context of speech synthesis. Specifically, we developed an unsupervised product-of-experts approach to annotate breath events in a corpus of spontaneous speech according to their salience. In combination with our bigram corpus method, this allowed us to create an automatically breathing TTS system with hard-to-predict breath events excluded, exhibiting a more fluent style. This spontaneous speech synthesiser automatically adjusts the prosodic realisation in accordance with the generated breathing style. In a listening test on two distinct spoken genres (one more, one less rehearsed), this synthesis was preferred over a baseline approach to automatic breathing treating all breath events the same.

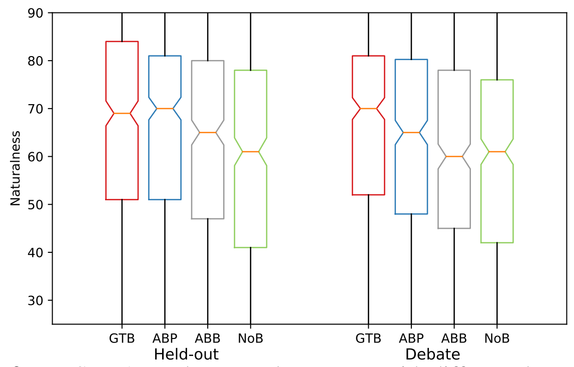

Fig. 2. MUSHRA results on spoken genres with different degrees of speech planning: impromptu (held-out) and rehearsed (debate).

6. REFERENCES

[1] S. Sundaram and S. Narayanan, "An empirical text transformation method for spontaneous speech synthesizers," in *Proc. Eurospeech*, 2003, pp. 1221–1224.

[2] S. Roekhaut, J.-P. Goldman, and A. C. Simon, "A model for varying speaking style in TTS systems," in *Proc. Speech Prosody*, 2010, pp. 096:1–096:4.

[3] N. Braunschweiler and L. Chen, "Automatic detection of inhalation breath pauses for improved pause modelling in HMM-TTS," in *Proc. SSW*, 2013, vol. 8, pp. 1–6.

[4] V. Klimkov, A. Nadolski, A. Moinet, B. Putrycz, R. Barra-Chicote, T. Merritt, and T. Drugman, "Phrase break prediction for long-form reading TTS: Exploiting text structure information," in *Proc. Interspeech*, 2017, pp. 1064–1068.

[5] D. H. Whalen, C. E. Hoequist, and S. M. Sheffert, "The effects of breath sounds on the perception of synthetic speech," *J. Acoust. Soc. Am.*, vol. 97, no. 5, pp. 3147–3153, 1995.

[6] U. Bernardet, S.-H. Kanq, A. Feng, S. DiPaola, and A. Shapiro, "Speech breathing in virtual humans: An interactive model and empirical study," in *Proc. VHCIE*, 2019, pp. 1–9.

[7] M. Włodarczak and M. Heldner, "Respiratory constraints in verbal and non-verbal communication," *Front. Psychol.*, vol. 8, pp. 708:1–708:11, 2017.

[8] A. L. Winkworth, P. J. Davis, E. Ellis, and R. D. Adams, "Variability and consistency in speech breathing during reading: Lung volumes, speech intensity, and linguistic factors," *J. Speech Lang. Hear. R.*, vol. 37, no. 3, pp. 535–556, 1994.

[9] Y.-T. Wang, J. R. Green, I. S. B. Nip, R. D. Kent, and J. F. Kent, "Breath group analysis for reading and spontaneous speech in healthy adults," *Folia Phoniat. Logo.*, vol. 62, no. 6, pp. 297–302, 2010.

[10] H. L. Mitchell, J. D. Hoit, and P. J. Watson, "Cognitive-linguistic demands and speech breathing," *J. Speech Lang. Hear. R.*, vol. 39, no. 1, pp. 93–104, 1996.

[11] A. Rochet-Capellan, G. Bailly, and S. Fuchs, "Is breathing sensitive to the communication partner?," in *Proc. Speech Prosody*, 2014, pp. 613–618.

[12] P. A. Barbosa and S. Madureira, "The interplay between speech and breathing across three Brazilian Portuguese speaking styles," in *Proc. Speech Prosody*, 2018, pp. 369–373.

[13] A. Henderson, F. Goldman-Eisler, and A. Skarbek, "Temporal patterns of cognitive activity and breath control in speech," *Lang. Speech*, vol. 8, no. 4, pp. 236–242, 1965.

[14] G. Bailly and C. Gouvernayre, "Pauses and respiratory markers of the structure of book reading," in *Proc. Interspeech*, 2012, pp. 2218–2221.

[15] É. Székely, G. E. Henter, J. Beskow, and J. Gustafson, "Spontaneous conversational speech synthesis from found data," in *Proc. Interspeech*, 2019, pp. 4435–4439.

[16] R. Clark, H. Silen, T. Kenter, and R. Leith, "Evaluating long-form text-to-speech: Comparing the ratings of sentences and paragraphs," in *Proc. SSW*, 2019, vol. 10, pp. 99–104.

[17] H. Zen, V. Dang, R. Clark, Y. Zhang, R. J. Weiss, Y. Jia, Z. Chen, and Y. Wu, "LibriTTS: A corpus derived from LibriSpeech for text-to-speech," in *Proc. Interspeech*, 2019, pp. 1526–1530.

[18] H. Pulakka, V. Myllyla, L. Laaksonen, and P. Alku, "Bandwidth extension of telephone speech using a filter bank implementation for highband mel spectrum," in *Proc. EUSIPCO*, 2010, pp. 979–983.

[19] É. Székely, G. E. Henter, and J. Gustafson, "Casting to corpus: Segmenting and selecting spontaneous dialogue for TTS with a CNN-LSTM speaker-dependent breath detector," in *Proc. ICASSP*, 2019, pp. 6925–6929.

[20] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Comput.*, vol. 14, no. 8, pp. 1771–1800, 2002.

[21] H. Zen, M. J. F. Gales, Y. Nankaku, and K. Tokuda, "Product of experts for statistical parametric speech synthesis," *IEEE T. Audio Speech Lang.*, vol. 20, no. 3, pp. 794–805, 2012.

[22] É. Székely, J. Kane, S. Scherer, C. Gobl, and J. Carson-Berndsen, "Detecting a targeted voice style in an audiobook using voice quality features," in *Proc. ICASSP*, 2012, pp. 4593–4596.

[23] É. Székely, G. E. Henter, J. Beskow, and J. Gustafson, "How to train your fillers: uh and um in spontaneous speech synthesis," in *Proc. SSW*, 2019, vol. 10, pp. 245–250.

[24] Y. Ferstl and R. McDonnell, "Investigating the use of recurrent motion modelling for speech gesture generation," in *Proc. IVA*, 2018, pp. 93–98.

[25] Google LLC, "Google Cloud Speech API video model," https://cloud.google.com/speech.

[26] R. M. Ochshorn and M. Hawkin, "Gentle forced aligner," https://github.com/lowerquality/gentle, 2017.

[27] R. Mama, "Tacotron-2 Tensorflow implementation," https://github.com/Rayhane-mamah/Tacotron-2, 2018.

[28] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, RJ Skerry-Ryan, R. A. Saurous, Y. Agiomyrgiannakis, and Y. Wu, "Natural TTS synthesis by conditioning WaveNet on mel spectrogram predictions," in *Proc. ICASSP*, 2018, pp. 4779–4783.

[29] D. Griffin and J. Lim, "Signal estimation from modified short-time Fourier transform," *IEEE T. Acoust. Speech*, vol. 32, no. 2, pp. 236–243, 1984.

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

[31] T. Lavee, M. Orbach, L. Kotlerman, Y. Kantor, S. Gretz, L. Dankin, S. Mirkin, M. Jacovi, Y. Bilu, R. Aharonov, and N. Slonim, "Towards effective rebuttal: Listening comprehension using corpus-wide claim mining," in *Proc. ArgMining*, 2019, pp. 58–66.

[32] M. Schoeffler, S. Bartoschek, F.-R. Stöter, M. Roess, S. Westphal, B. Edler, and J. Herre, "WebMUSHRA—A comprehensive framework for web-based listening tests," *J. Open Res. Softw.*, vol. 6, no. 1, 2018.