HOTWORD CLEANER: DUAL-MICROPHONE ADAPTIVE NOISE CANCELLATION WITH DEFERRED FILTER COEFFICIENTS FOR ROBUST KEYWORD SPOTTING

Yiteng (Arden) Huang, Turaj Z. Shabestary, Alexander Gruenstein

Google Inc., USA

{ardenhuang, turajs, alexgru}@google.com

ABSTRACT

This paper presents a novel dual-microphone speech enhancement algorithm to improve noise robustness of hotword (wake-word) detection as a special application of keyword spotting. It exploits two unique properties of hotwords: they are leading phrases of valid voice queries that we intend to respond and have short durations. Consequently an STFT-based adaptive noise cancellation method modified to use deferred filter coefficients is proposed to extract hotwords out from noisy stereo microphone signals. The new algorithm is tested with two considerably different neural hotword detectors. Both systems have significantly reduced the false-reject rate when background has strong TV noise.

Index Terms— Keyword spotting and hotword/wake-word detection, dual-microphone noise reduction, adaptive filter with deferred coefficients, smart speech enhancement, microphone array processing for machine learning

1. INTRODUCTION

Voice-enabled interfaces are no longer futurist speculation. Recently they have evolved into strategic solutions for enhanced user experience and new business opportunities, as exemplified by Google Assistant and Amazon Alexa. These solutions all aim to deliver full hands-free interaction by using a keyword spotting (KWS) method to control voice input. The KWS algorithm continuously monitors a stream of audio for a preset phrase (i.e., keyword like "Ok/Hey Google" and "Alexa") and activates speech recognition and comprehension upon detecting it.

KWS systems run on client devices, such as mobile phones or smart speakers, so it's critical that they run at low latency with small memory and compute footprints. Early dynamic time warping (DTW) based template matching methods have some difficulties to meet these prerequisites and their performance was quite poor [1, 2]. It was the advent of the hidden Markov model (HMM) that made KWS an increasingly popular topic for research [3, 4, 5, 6]. So far the arguably most exciting advance in this field was driven by deep learning and big data [7]. Many successful systems have been reportedly built on a variety of neural network (NN) structures, including deep neural network (DNN) [8], convolutional neural network (CNN) [9], deep residual network (ResNet) [10], recurrent neural network (RNN) [11, 12], long short-time memory (LSTM) [13, 14].

Voice assistants must be usable in a wide range of acoustic conditions. Multi-style training (MTR) has been useful to fight far-field distortions and moderate non-speech noise [15]. But speech-like interference (e.g., TV noise) remains a challenging issue. We have also investigated a number of beamforming and multichannel Wiener filtering algorithms. Although helpful, they can produce only marginal improvement with a small array of two microphones. It is noteworthy that some promising results were obtained with the so-called keyword sifter [16]. This new approach complements a KWS system with a top-down mechanism such that selective attention (aka the cocktail-party effect [17]) can be mimicked. But the sifter needs to run the KWS classifier twice in succession, potentially increasing latency and CPU usage. Moreover, an additional parameter, the threshold at which to activate the sifter, must be tuned via experimentation and it is impossible to have one value that fits all signal-to-noise ratios (SNRs).

In this paper, we address hotword detection (Google's term for wake-word detection) as a special application of KWS. KWS is regarded as a more general-purpose task in which a keyword can take place anytime anywhere, while in hotword detection a hotword always precedes a voice query. By exploiting this unique property, we have developed another effective yet much simpler speech enhancement algorithm for hotword detection on devices with two microphones.

2. HOTWORD CLEANER

When we use two microphones that are separated by a small distance to capture sound from surrounding speakers and ambient noise sources, the two microphone signals are coherent. If an adaptive filter is applied to one microphone signal and the result is subtracted from the other microphone signal like in adaptive noise cancellation (ANC) [18], the residual error would have smaller magnitudes. But this inevitably cancels not only noise but also speech. Since speech and noise signals presumably maintain different coherence levels at the two microphones, a useful adaptive filter for noise cancellation is the one that compensates only the noise coherence. This requires a mechanism to supervise the operation of adaptive filtering: ideally, adaptation should be performed when speech is absent and must be halted otherwise. The keyword sifter [16] relies on a neural KWS classifier to compute such a speech/noise mask. In this paper, we propose a simple alternative, hotword cleaner, which is completely based on signal processing.

The hotword cleaner makes the following two assumptions:

1) A hotword is the *leading* phrase of a valid voice query, which implies that the short segment immediately preceding a hotword contains no speech from the hotword speaker but only noise or other competing speaker's speech.
2) A hotword has a short duration, typically less than 1 s.

As illustrated by Fig. 1, the cleaner consists of two processing layers, one on top of the other. The bottom layer works like a traditional ANC system that adjusts the filter coefficients repeatedly to minimize the mean square of the prediction error $e(n)$ between the two microphone signals. But we don't take $e(n)$ as the cleaner's

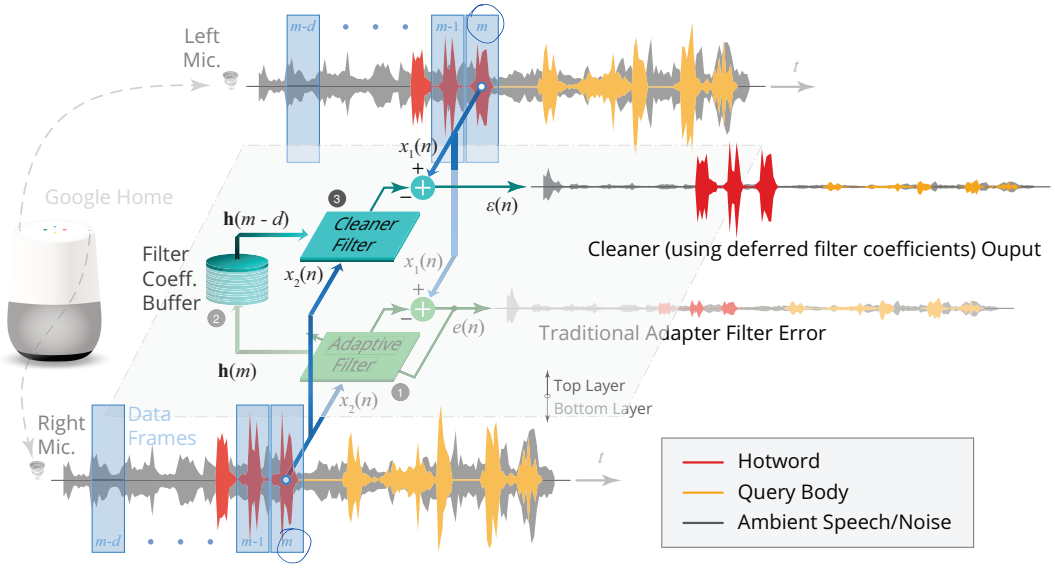

Fig. 1: Illustration of dual-microphone hotword cleaner using an adaptive noise cancellation filter with deferred filter coefficients.

output. The bottom layer saves the estimated filter coefficients into a first-in-first-output (FIFO) buffer. In the top layer, the buffer output which is a set of filter coefficients with a delay of d frames is used to process the two microphone signals and produce the cleaned signal output $\varepsilon(n)$.

Let's briefly explain why the cleaner is able to achieve the goal of speech enhancement. At the leading edge of a hotword, the filter coefficients used in the top layer are estimated d frames before the current time where, according to the first assumption above, the person saying the hotword wasn't speaking yet. This means that the filter coefficients don't compensate the hotword coherence across the two microphones, so only the noise is canceled while the hotword is retained in the cleaner's output. Thanks to the second assumption we made above, and because hotwords are usually short, it is straightforward to select a value for d such that the time span of d frames is almost always greater than the duration of the hotword. As a result, even by the time we come to the trailing end of the hotword, the coefficients popped out from the buffer have *not* been affected by the hotword and signal cancellation won't occur.

Similar to the keyword sifter, an important design parameter for the cleaner is d, i.e., the buffer length for filter coefficients. If d is much smaller than the length of hotwords, many filtered hotword samples at the cleaner's output will experience severe signal cancellation around their tails, leading to low confidence in recognizing their last several phones. But, on the contrary, if d is too large, cross-microphone noise coherence may have significantly changed over such a period of time. Using the deferred filter coefficients can only achieve a low gain in noise reduction. In our research for detection of "Ok Google" and "Hey Google", d is set to a value with an equivalent delay of 768 ms.

3. FAST-RLS WITH DEFERRED FILTER COEFFICIENTS

Generally speaking, any adaptive filters can be modified to allow for deferred filter coefficients. As an example, Table 1 presents the modifications we made to the fast recursive least squares (RLS) algorithm in the short-time Fourier transform (STFT) domain (see for comparison Table 1 of [16]). Since we have followed the same notation used in [16], the detailed definition of variables is not repeated here for brevity of presentation.

Table 1: Summary of the STFT-domain fast RLS algorithm with deferred filter coefficients.

Parameters:

d = size of the delay buffer in number of frames

L = filter length, λ = forgetting factor

δ = coefficient to initialize $\mathbf{P}(0)$

Initialization:

$\mathbf{h}(0) = \mathbf{0}$, $\mathbf{x}_2(0) = \mathbf{0}$,

$\mathbf{P}(0) = \delta^{-1}\mathbf{I}$, where $\mathbf{I}$ is the identity matrix of rank L.

Adaptation: for frames $m = 1, 2, \cdots$

A-priori error:

$$E(m) = X_1(m) - \mathbf{h}^H(m-1)\mathbf{x}_2(m),$$

Kalman gain vector:

$$\mathbf{g}(m) = \frac{\mathbf{P}(m-1)\mathbf{x}_2(m)}{\lambda + \mathbf{x}_2^H(m)\mathbf{P}(m-1)\mathbf{x}_2(m)},$$

Update:

$$\mathbf{P}(m) = \lambda^{-1}\left[\mathbf{P}(m-1) - \mathbf{g}(m)\mathbf{x}_2^H(m)\mathbf{P}(m-1)\right],$$

$$\mathbf{h}(m) = \mathbf{h}(m-1) + \mathbf{g}(m)E^*(m),$$

Output:

$$\mathcal{E}(m) = X_1(m) - \mathbf{h}^H(m-d)\mathbf{x}_2(m).$$

4. NEURAL DETECTORS

It is widely known that speech enhancement algorithms, while improving SNRs, can introduce distortion. Some hotword detectors can be more sensitive than others to speech distortion. So, we tested the proposed hotword cleaner with two detectors that have considerably different architectures. They have the same frontend feature extractor but differ in acoustic model (AM) and decoder. In the first detector, AM is accomplished by a CNN and the decoder simply searches for a best path of the hotword's phoneme sequence. The second detector has an end-to-end (E2E) structure and uses NNs for both AM and decoder [19].

For feature extraction, a single-channel audio signal is first segmented into 25 ms long frames with 10 ms hops. For each frame, 40-

Table 2: Comparison of the CNN and E2E models in number of parameters and multiply-accumulate (MAC) operations.

Model	#Params	MAC/10ms
CNN	1.71M	1.86M
E2E	0.32M	0.16M

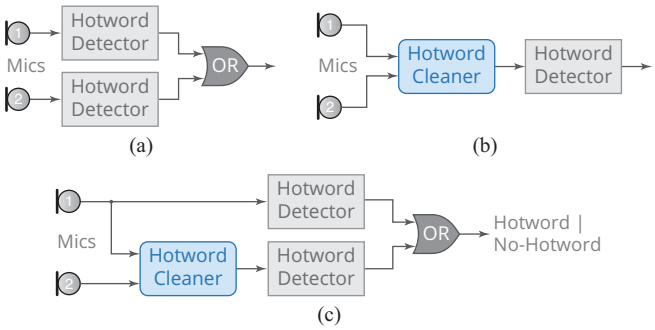

Fig. 2: Three strategies for integrating hotword cleaner and detector: (a) *baseline*, (b) *cleaner-only*, and (c) *hybrid*.

channel log-mel-filter-bank energies (LMFBEs) are computed and normalized using per-channel energy normalization (PCEN) [20].

- The CNN model takes stacked LMFBE feature vectors as inputs, with 24 left and 15 right context frames. The convolutional layer sweeps 92 non-overlapping 8×8 patches across time and frequency and generates $92 \times (40/8) \times (40/8) = 2300$ outputs. The following three hidden layers are fully-connected dense layers with rectified linear unit (ReLU) and producing 512 outputs. The last layer is also a dense layer but the activation function is softmax. The outputs are posterior probabilities of 7 phone classes (6 phones in "Ok/Hey Google" and one for others including silence).
- The E2E model runs in streaming mode. Its input feature vector consists of only 3 LMFBE frames with 1 left and 1 right context frame. To make the model smaller, SVDF layers are deployed [21]. The AM NN contains 4 such layers and each is followed by a bottleneck layer with linear activation function. The last AM layer also outputs posterior probabilities of 7 phone classes. The decoder NN has 3 SVDF layers but no bottleneck layers. The last layer is a dense layer using softmax to produce the posterior probability of finding a hotword. Note that the two NNs were trained successively rather than jointly [19].

Table 2 presents a comparison of the CNN and E2E models in terms of size and arithmetic complexity: the E2E model is more than 5x smaller and 11-fold faster than the CNN model.

5. SYSTEM INTEGRATION STRATEGIES

For a two-microphone audio system, if hotword detection can be performed individually on each microphone signal, it is like asking two listeners to monitor the same sound field collaboratively: a hotword is deemed identified if any of the listeners identifies one. Such an approach can reduce false rejects (FRs) but can also increase false accepts (FAs). So a pair of detectors of roughly equal accuracy that work collectively may not significantly outperform each of the detectors. But if they have diversified yet complementary strengths/weaknesses, we can benefit from a well-thought-out integration strategy.

Figure 2 portrays three strategies for cleaner system integration. The *baseline* strategy is a two-microphone system which computes logical-or of the two independent hotword detection results on each

Table 3: Summary of collected data for performance evaluation.

Dataset	Number of Utterances	Length (hours)
Far-Field Clean	6,274	29.1
TV Noise	5,694	27.5
Negative	55,469	1,175.1

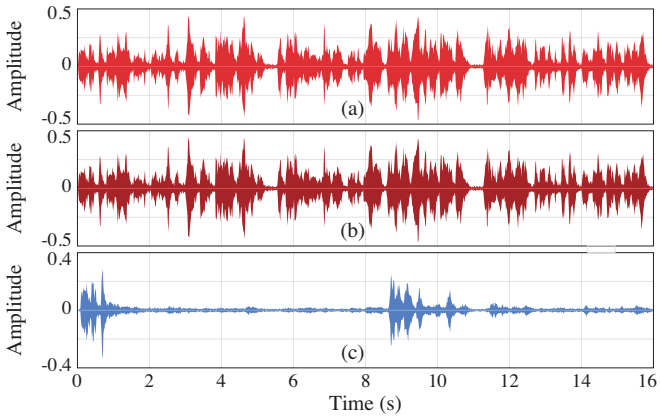

Fig. 3: Sample stereo utterance with TV noise processed by the hotword cleaner: (a) the left microphone signal, (b) the right microphone signal, and (c) the cleaned signal.

input audio channel. The *cleaner-only* strategy runs detection only on the cleaner output. Finally, the *hybrid* strategy takes the logical-or of hotword detection on the cleaned audio and on one channel of the original audio.

While the *cleaner-only* strategy is typically how speech enhancement algorithms are applied, there is a key advantage to the *hybrid* strategy: latency. Applying the cleaner adds a significant lag to the audio stream of 128 ms (STFT window size). In the *cleaner-only* strategy, all hotword detections will suffer from this increased latency. However, in the *hybrid* strategy, there need not be any additional latency in cases where the hotword detector triggers on the non-cleaned audio channel. Thus, the hotword cleaner will add detection latency *only on utterances where hotword detection would have failed without it*. The second advantage of the hybrid strategy is the corner case where a user is talking before he says the hotword. In this case, hotword detection is likely to fail on the cleaned channel but succeed on the uncleaned.

On the other hand, the potential advantages of the *cleaner-only* approach are that (1) the cleaner can reduce false alarms due to noise sources such as TVs, and (2) only a single instance of the hotword detector must be run.

6. EXPERIMENTS

6.1. Experimental Setup

We use real re-recorded data to evaluate the proposed hotword cleaner. The experimental setup is similar to that described in [16], but this study includes a larger set of data with longer preambles. Due to the limitation of space, positive datasets consider only two conditions: far-field clean and TV noise in the background. The negative dataset is composed of television audio recorded on a far-field Google Home. Table 3 gives a summary of the data.

Again, NNs were implemented using Google's TensorFlow™ library [22] and trained with deep learning algorithms on both logs and collected data from gender-balanced pool of volunteers with a

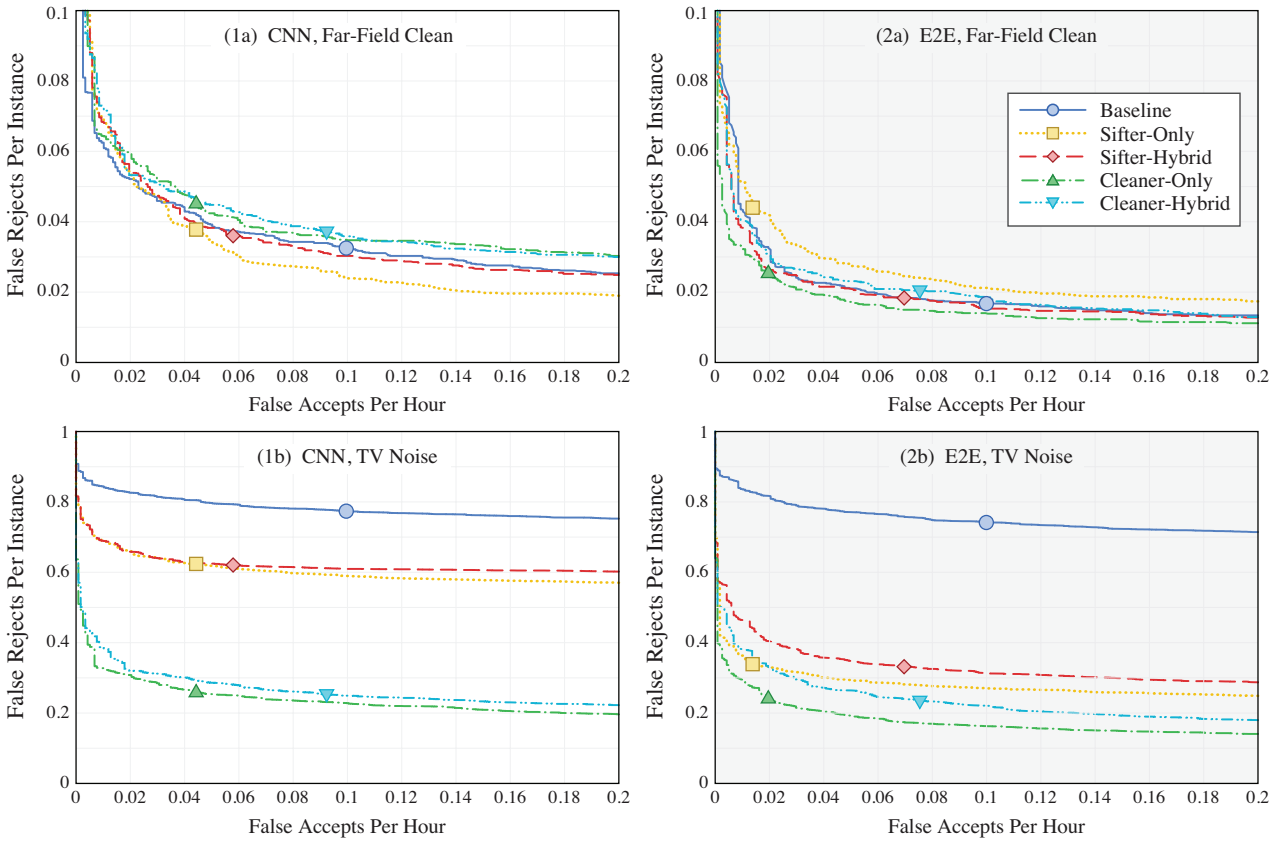

Fig. 4: ROCs comparing performance of the keyword sifter and hotword cleaner against their baseline systems working with CNN and E2E models on far-field clean and TV-noise datasets. Note that for the systems using the same NN model marker positions correspond to the same threshold.

variety of accents [23].

In this paper, we have used the following parameters for the hotword cleaner: 16 kHz sampling rate, 128 ms windows with 50% overlap, $d = 12$, $L = 3$, $\lambda = 0.993$, and $\delta = 0.1$. For the keyword sifter, the parameters are the same as that used in [16] – in particular the near-trigger threshold is set to 1% of the NN model's threshold.

6.2. Cleaned Sample Utterance

First, we would like to present a sample utterance processed by the hotword cleaner to give the reader an intuitive feel for how the cleaner helps fight background TV noise. As shown in Fig. 3(a) and (b), the utterance lasts for 16 s and is heavily contaminated by TV noise. The hotword appears around 8.5 s and the input SNR is about 0 dB. Figure 3(c) plots the cleaned signal. Simply by visual examination, it is evident that the hotword has stood out and the output SNR has greatly improved. It is interesting to note that the cleaner follows an acoustic scene change with a lag of about 1.5 s.

6.3. Receiver Operating Curves (ROCs)

For hotword detection, the most informative performance measure is a ROC, which visualizes the FR rate (in number of FRs per instance) as a function of the FA rate (in number of FAs per hour). Figure 4 presents the ROCs of the 5 tested systems. The baseline is after what was described by Fig. 2(a). The cleaner-only and sifter-only systems follow the strategy of Fig. 2(b). The cleaner-hybrid and sifter-hybrid systems follow the strategy of Fig. 2(c). On the far-field clean dataset, the performance of either model is comparable with and without the hotword cleaner. But the cleaner remarkably reduces the FR rate on the TV-noise dataset: a relative improvement of 66% with both models at the operating point of a fixed threshold. The cleaner also outperforms the keyword sifter in these experiments. The performance degradation caused by using the hybrid strategy over the cleaner-only strategy on these datasets is merely marginal.

7. CONCLUSIONS

In this paper we have presented a dual-microphone speech enhancement algorithm code-named as hotword cleaner for robust hotword detection in noisy environments. It takes advantage of two unique properties associated with hotwords: hotwords are always leading phrases in valid voice queries and hotwords are short in duration. So a STFT-based adaptive noise cancellation algorithm with deferred filter coefficients was proposed to extract hotwords from noisy stereo microphone signals. The hotword cleaner was tested with two detectors which have considerably different architectures: CNN and E2E. In both cases, cleaner reduced the FR rate by over 66% relative on re-recorded utterances with strong TV background noise.

8. ACKNOWLEDGMENTS

The authors would like to thank Thad Hughes and Taylor Applebaum for their support and involvement in the early stage of this research.

Keyword spotting classifier

9. REFERENCES

[1] J. S. Bridle, "An efficient elastic-template method for detecting given words in running speech," *Brit. Acoust. Soc. Meeting*, pp. 1–4, Apr. 1973.

[2] A. L. Higgins and R. E. Wohlford, "Keyword recognition using template concatenation," in *Proc. IEEE ICASSP*, 1985, pp. 1233–1236.

[3] R. C. Rose and D. B. Paul, "A hidden Markov model based keyword recognition system," in *Proc. IEEE ICASSP*, 1990, pp. 129–132.

[4] J. R. Rohlicek, W. Russell, S. Roukos, and H. Gish, "Continuous hidden Markov modeling for speaker-independent wordspotting," in *Proc. IEEE ICASSP*, 1990, pp. 627–630.

[5] J. G. Wilpon, L. R. Rabiner, C.-H. Lee, and E. Goldman, "Automatic recognition of keywords in unconstrained speech using hidden Markov models," *EEE Trans. Acoust., Speech, Signal Process.*, vol. 38, pp. 1870–1878, Nov. 1990.

[6] J. G. Wilpon, L. G. Miller, and P. Modi, "Improvements and applications for key word recognition using hidden Markov modeling techniques," in *Proc. IEEE ICASSP*, 1991, pp. 309–312.

[7] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, September 2012.

[8] G. Chen, C. Parada, and G. Heigold, "Small-footprint keyword spotting using deep neural networks," in *Proc. IEEE ICASSP*, 2014, pp. 4087–4091.

[9] T. N. Sainath and C. Parada, "Convolutional neural networks for small-footprint keyword spotting," in *Proc. InterSpeech*, 2015, pp. 1478–1482.

[10] R. Tang and J. Lin, "Deep residual learning for small-footprint keyword spotting," in *Proc. IEEE ICASSP*, 2018, pp. 5484–5488.

[11] S. Fernández, A. Graves, and J. Schmidhuber, "An application of recurrent neural networks to discriminative keyword spotting," in *Proc. Int. Conf. Artificial Neural Networks*, 2007, pp. 220–229.

[12] P. Baljekar, J. F. Lehman, and R. Singh, "Online word-spotting in continuous speech with recurrent neural networks," in *Proc. Spoken Language Technology Workshop (SLT)*, 2014, pp. 536–541.

[13] M. Wöllmer, B. Schuller, and G. Rigoll, "Keyword spotting exploiting long short-term memory," *Speech Commun.*, vol. 55, pp. 252–265, February 2013.

[14] M. Sun, A. Raju, G. Tucker, S. Panchapagesan, G. Fu, A. Mandal, S. Matsoukas, N. Strom, and S. Vitaladevuni, "Max-pooling loss training of long short-term memory networks for small-footprint keyword spotting," in *Proc. Spoken Language Technology Workshop (SLT)*, 2016, pp. 474–480.

[15] C. Kim, A. Misra, K. Chin, T. Hughes, A. Narayanan, T. Sainath, and M. Bacchiani, "Generation of large-scale simulated utterances in virtual rooms to train deep-neural networks for far-field speech recognition in Google Home," in *Proc. InterSpeech*, 2017, pp. 379–383.

[16] Y. Huang, T. Hughes, T. Z. Shabestary, and T. Applebaum, "Supervised noise reduction for multichannel keyword spotting," in *Proc. IEEE ICASSP*, 2018, pp. 5474–5478.

[17] E. C. Cherry, "Some experiments on the recognition of speech, with one and with two ears," *J. Acoust. Soc. Am.*, vol. 25, pp. 975–979, Sept. 1953.

[18] B. Widrow, J. R. Glover, J. M. McCool, J. Kaunitz, C. S. Williams, R. H. Hearn, J. R. Zeidler, E. Dong, and R. C. Goodlin, "Adaptive noise cancelling: principles and applications," *Proc. IEEE*, vol. 63, pp. 1692–1716, Dec. 1975.

[19] R. Alvarez and H.-J. Park, "End-to-end streaming keyword spotting," IEEE ICASSP, 2019, submitted for publication.

[20] Y. Wang, P. Getreuer, T. Hughes, R. F. Lyon, and R. A. Saurous, "Trainable frontend for robust and far-field keyword spotting," in *Proc. IEEE ICASSP*, 2017, pp. 5670–5674.

[21] P. Nakkiran, R. Alvarez, R. Prabhavalkar, and C. Parada, "Compressing deep neural networks using a rank-constrained topology," in *Proc. InterSpeech*, 2015, pp. 1473–1477.

[22] Google Inc., "TensorFlow™: An open-source library for Machine Intelligence," `https://www.tensorflow.org`, [Online; Latest Accessed 20-Oct-2017].

[23] T. Hughes, K. Nakajima, L. Ha, A. Vasu, P. Moreno, and M. LeBeau, "Building transcribed speech corpora quickly and cheaply for many languages," in *Proc. InterSpeech*, 2010, pp. 1914–1917.