Region Proposal Network Based Small-Footprint Keyword Spotting

Jingyong Hou, *Student Member, IEEE*, Yangyang Shi, *Senior Member, IEEE*,
Mari Ostendorf, *Fellow, IEEE*, Mei-Yuh Hwang, *Fellow, IEEE*, Lei Xie, *Senior Member, IEEE*

***Abstract*—We apply an anchor-based region proposal network (RPN) for end-to-end keyword spotting (KWS). RPNs have been widely used for object detection in image and video processing; here, it is used to jointly model keyword classification and localization. The method proposes several anchors as rough locations of the keyword in an utterance and jointly learns classification and transformation to the ground truth region for each positive anchor. Additionally, we extend the keyword/non-keyword binary classification to detect multiple keywords. We verify our proposed method on a hotword detection data set with two hotwords. At a false alarm rate of one per hour, our method achieved more than 15% relative reduction in false rejection of the two keywords over multiple recent baselines. In addition, our method predicts the location of the keyword with over 90% overlap, which can be important for many applications.**

***Index Terms*—Keyword spotting, Region proposal network, Multi-task learning, End-to-end, False rejection, False alarm.**

I. Introduction

SPEECH keyword spotting (KWS) is the task of detecting keyword(s) from streaming audio or a pre-recorded audio utterance. It plays an important role in applications such as audio search, automatic detection of profanity in live broadcasts, and voice control of devices. Applications such as audio search that require handling open vocabularies typically build on large vocabulary speech recognition (LVCSR) technology, e.g. [1]–[3]. In contrast, device control often involves low-resource scenarios (limited memory and computation) that require a small-footprint implementation. An important example is wake-up word (or hotword) detection, widely used in virtual assistants for smart phones, personal computers, smart speakers, etc. In this paper, we focus on the small-footprint wake-up word detection scenario.

The dominant approach to small-footprint KWS was the keyword-filler Hidden Markov Model (HMM) [4]–[8]. One HMM was used for predefined keywords, often based on a phone sequence, and a second filler HMM scored non-keyword speech segments, usually via a phone loop. Gaussian mixture models (GMMs) were once used to model the observation features of the keyword/filler HMMs, but now KWS systems often replace GMMs with deep neural networks (DNNs) to provide HMM state posteriors [9]–[13]. Both the HMM and hybrid HMM-DNN approaches identify keyword and non-keyword regions with a sequential decoding process. In [14], it was shown that better performance could be obtained by post-processing the sequence of posteriors from a feedforward DNN with subword and filler outputs, training with an objective that associated all frames in a subword or filler with the corresponding label. Subsequent work with variants of this approach [15]–[17] confirmed the finding that feedforward DNNs gave significant improvement over the HMM-DNN approach in low-footprint scenarios.

Further performance gains have been obtained with architectures that treat a keyword or key-phrase as a single modeling unit and simply detect its presence in a segment of speech, which facilitates end-to-end training with keyword hit accuracy as an objective. Some configurations use a sequence model [18]–[20]. Other configurations assume a trigger position in the word, either automatically learned via max pooling [21] or attention [22], or specified as the last frames of the keyword [23]–[25]. These models give good detection accuracy, but they do not provide the precise keyword location. However, accurate location of the keyword can provide valuable information for downstream modules. In a low-resource scenario, on-device processing first detects the keyword, then audio is sent to a server for keyword verification and command or query recognition. Accurate keyword location facilitates verification and ensures that the full user utterance is captured for recognition. In addition, we hypothesize that learning to predict the accurate locations of keywords will improve the keyword detection accuracy. Therefore in this paper, we propose to jointly optimize keyword detection and location. in an end-to-end multi-task learning fashion.

To jointly optimize keyword detection and location prediction, we adapt the region proposal network (RPN) for KWS. The RPN was first proposed in [26] for object detection, where it achieved cutting edge performance by jointly optimizing object classification and location estimation. Different from object detection, where RPNs are used to deal with static images, in this paper the RPN is re-designed to process streaming audio signals. Experimental results show that, at a fixed false alarm rate of one per hour, the RPN method achieves more than 15% relative reduction in false rejection rates (FRR) for two keywords over other end-to-end approaches. To the best of our knowledge, this is the first successful work of applying RPN to small-footprint keyword spotting. Source code is available at: https://github.com/jingyonghou/RPN_KWS.git.

II. RPN-based KWS

As shown in Fig. 1, the proposed system consists of two modules: a feature extractor and the RPN.

Jingyong Hou and Lei Xie are with School of Computer Science and Engineering, Northwestern Polytechnical University, Xi'an, China.

Yangyang Shi and Mei-Yuh Hwang are with Mobvoi AI Lab, Seattle, USA

Mari Ostendorf is with Electrical & Computer Engineering, University of Washington, Seattle, USA

A. The feature extractor M_0

The feature extractor module takes T frames of raw speech features $\mathbf{x} = (x_1, x_2, ..., x_T)$ as input and outputs $\mathbf{h} = (h_1, h_2, ..., h_T)$ as the high-level feature representation of the input speech: $\mathbf{h} = M_0(\mathbf{x}; \theta_0)$, where θ_0 is the parameters of model M_0. In this paper, we use gated recurrent units (GRU) as the feature extractor.

B. Region proposal network M_1 and M_2

Rather than score all possible spans, the RPN approach selects a small number of anchors (candidate time regions for containing a keyword) with sub-network M_1 and predicts the transformation needed to extract the full keyword associated with that region using sub-network M_2.

1) Anchors: For each frame t, K anchors are proposed, all with an end point at t, but with varying starting points that are uniformly spaced over the allowable lengths of the keyword. Here K is a hyper parameter which will be tuned on a development data set.

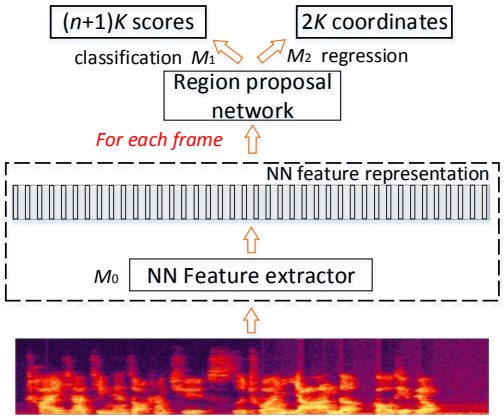

Fig. 1. Framework of RPN-based KWS. This figure shows K anchors are proposed at each time frame, with n keywords to identify. For the classification network M_1, the output dimension is $(n+1)$ per anchor. The regression network M_2 transforms each positive anchor to the keyword location and hence the output dimension is 2 (shift and scale) per anchor.

2) Region classification: The sub-network M_1 is used to classify anchors to a keyword ID. M_1 takes each frame h_t from the feature extractor as input and outputs one posterior vector for each of the K anchors: $\overrightarrow{y_t} = M_1(h_t; \theta_1)$, where $\overrightarrow{y_t} = (y_t^1, y_t^2, ..., y_t^K)$ is the sequence of posterior vectors corresponding to the K anchors at time t. For hotword detection with only one keyword, only a scalar keyword posterior is needed. Here, the RPN is used to handle n keywords that cannot be simultaneously spoken. Therefore, $|y_t^i| = n+1$ and y_t^i is computed using a softmax.

3) Region Transformation: The sub-network M_2 is used to learn a time transformation for each positive anchor to the ground-truth location of the keyword. M_2 also takes h_t as input and outputs one transformation vector for each anchor: $\overrightarrow{p_t} = M_2(h_t; \theta_2)$, where $\overrightarrow{p_t} = (p_t^1, p_t^2, ..., p_t^K)$ is the sequence of transformation vectors corresponding to K anchors. Each p_t^i consists of a shifting factor and a scaling factor.

Suppose we have two regions: anchor $P = (t_1, t_2)$ and ground truth box of keyword $Q = (t_3, t_4)$, where notation $A = (b, e)$ means the region A starts from frame b and ends at frame e. In order to transform P to Q, we first shift P by u to make its midpoint coincide with the midpoint of Q, and then scale the shifted P by v (fixing the midpoint) to make it the same length as Q, where

$$\begin{aligned} u &= (t_3 + t_4)/2 - (t_1 + t_2)/2, \\ v &= (t_4 - t_3)/(t_2 - t_1). \end{aligned}$$

In practice, instead of predicting shift u and scale factor v directly, M_2 predicts the normalized shift $\hat{u} = u/l$ and the logarithm of the scale $\hat{v} = \log(v)$, where $l = t_2 - t_1$ is the length of the corresponding anchor, The normalized $\hat{u}$ and $\hat{v}$ are used in the mean-squared error (MSE) loss function in training, similar to the work in object recognition. The normalized $\hat{u}$ generalizes across lengths. Using $\hat{v}$ values that are free to range from negative to positive numbers simplifies training [26].

4) Region proximity by IoU: We use the Intersection over union (IoU) ratio of overlap between two regions to measure their proximity. Specifically the proximity between anchor $P = (t_1, t_2)$ and ground truth $Q = (t_3, t_4)$ is calculated as the ratio of intersection over union as follows:

$$\begin{aligned} \text{IoU}(P, Q) &= \frac{P \cap Q}{P \cup Q}, \\ P \cap Q &= \max(\min(t_2, t_4) - \max(t_1, t_3), 0), \\ P \cup Q &= (t_4 - t_3) + (t_2 - t_1) - (P \cap Q). \end{aligned} \tag{1}$$

IoU is used to decide if an anchor is a positive training sample for a certain keyword, as explained next.

5) Selection of training samples: To train classification model M_1, we assign a category label to each anchor. For negative training utterances (those that do not contain any keyword), all anchors are negative anchors, corresponding to category label 0. For a positive training utterance, we calculate IoU between each anchor and the ground truth region of the keyword. If IoU is bigger than 0.7, this anchor will be treated as a positive anchor and be assigned a category label $\in 1..n$ corresponding to the keyword ID. If IoU is smaller than 0.3, this anchor will be used as a negative anchor and is assigned with the category label 0. Any anchor with IoU between 0.3 and 0.7 is discarded as ambiguous, i.e. not used in training, so not all K anchors will be used for any given time t. Only positive anchors participate in back propagation of M_2.

The number of negative anchors is usually much more than the number of positive anchors in each utterance. A down-sampling strategy is applied during back propagation. Specifically, for each utterance, 100 anchors are selected for back propagation training. Among them, at most 50 anchors are randomly selected from positive anchors, while the rest are randomly selected from negative anchors.

The selected training anchors $a(i)$, associated posteriors $y(i)$ and transformations $p(i) = (\hat{u}(i), \hat{v}(i))$ from all utterances are denoted as $\{a(i)\}$, $\{y(i)\}$ and $\{p(i)\}$, respectively, $i = 1, \ldots, N$. Let $\mathcal{A}^+$ and $\mathcal{A}^-$ denote the positive and negative anchor subsets, respectively.

6) Loss function: We minimize an MTL objective function, with the loss function:

$$Loss = \frac{1}{N}\sum_{i=1}^{N} L_c(y(i), y^*(i)) + \frac{\lambda}{N_+}\sum_{i:a(i)\in\mathcal{A}^+} L_r(p(i), p^*(i)). \quad (2)$$

where L_c is the cross-entropy loss between the predicted probability $y(i)$ and the ground-truth label $y^*(i)$ for anchor i, and L_r is the MSE loss between the predicted transformation vector $p(i)$ and its training target $p^*(i)$. All N selected anchors (positive and negative) are used to train the classifier M_1; only positive anchors ($N^+ = |\mathcal{A}^+|$) are used to train the regression model M_2. The two losses are normalized by the respective training sizes N and N_+, weighted by λ.

7) Streaming inference: When processing streaming audio, the RPN predicts K n-dimensional posterior classification vectors $\overrightarrow{y_t} = (y_t^1, y_t^2, ..., y_t^K)$ for each frame t. For each keyword $j \in 1, \ldots, n$, we find the anchor $a_t(j)$ that has the maximum posterior value:

$$a_t(j) = \text{argmax}_{k\in(1..K)} y_t^k(j).$$

If $y_t^{a_t(j)}(j) > \gamma_j$, then we say keyword j is triggered, where γ_j is a confidence threshold tuned on the development set with a given false alarm rate (FAR) requirement.[1] Once a keyword is triggered, any hits within the next second are ignored.

III. Experiments

A. Corpus

A corpus of wake-up words collected from a commercial smart speaker is used in our experiment. This data set is called *Ticmini2*. The data set has about 225 hours of data, in about 255k utterances. It is collected from 788 subjects, ages 3-65. Keyword and non-keyword data with different distances from the smart speaker (1, 3 and 5 meters) are collected from each subject. Different noises (typical home environment noises like music and TV) with varying signal-to-noise (SNR) ratios are played in the background during the collection. There are about 187 hours of non-keyword utterances; the rest contains either one 'Hi Xiaowen' keyword or one 'Nihao Wenwen' keyword per utterance. We randomly divide the above data set into training, development and testing sets by different speakers. The detailed information of this data set can be found in Table I. An HMM and time delay neural network acoustic model (AM) trained with general speech data is used to obtain the starting and ending points of the keywords.

TABLE I
CORPUS STATISTICS (#SPEAKERS/#UTTERANCES)

Data set	Train (60%)	Dev (10%)	Test (30%)
Hi Xiaowen	474/ 21,825	78/ 3,680	236/10,641
Nihao Wenwen	474/ 21,800	78/ 3,677	236/10,641
Non-keyword	418/113,898	67/17,522	203/51,613
All	474/157,523	78/24,879	236/72,895

B. Setup

1) Proposed method: For the feature extractor M_0, 2-layers of unidirectional GRU and an output projection layer with ReLU activation are used. Each GRU layer has 128 cells. The projection layer also has 128 output nodes. 40-dimensional Mel-filterbank features, with 25ms frame length and 10ms frame shift, are used as input to M_0.

By counting the duration of keywords in the training set, we find that over 99.9% of keywords are between 30 frames and 220 frames. So, for each frame t, we select K anchors with their lengths uniformly distributed from $30 - 220$ frames, all with frame t as the ending frame. For the number of anchors K, we have tried 7, 10, 13, 16 and 20. Since $K = 20$ gave only a small gain over $K = 16$, higher values were deemed unnecessary. $K = 20$ is used for all later experiments.

For M_1 and M_2 of the RPN, two linear layers are adopted. Taking the output of M_0 as input, M_1 outputs 3 classification probabilities for each training anchor, and M_2 outputs the 2 transformation factors for each positive training anchor.

For neural network training with ADAM optimization, we tried different batch sizes (200, 400 utterances) and learning rates (0.0005, 0.001, 0.002, 0.003). The batch size 400 and learning rate 0.002 were the optimal on the development set.

We investigated the effect of weight λ over the regression loss L_r in Eq. 2. We tried $\lambda = 0, 1, 2, 3, 4, 5$ and concluded that $\lambda = 3$ is the optimum on the development set.

2) Baselines: Three baseline systems are implemented in this paper. One is based on the Deep KWS [14] that is a widely used baseline [15], [16], [22], [24]. To detect the predefined keywords, it predicts 5 Chinese syllables ('Hi', 'Xiao', 'Wen', 'Ni' and 'Hao') and a 'filler' which represent non-keyword frames. (Using whole words gave worse results.) To improve performance, the current frame is spliced together with 15 history frames and 5 future frames as input to feature extractor. The smoothing window and sliding window are set to 20 and 100, respectively. The second baseline is RNN-attention [22], which automatically learns the trigger position. A sliding window of 220 frames is used in testing, consistent with the maximum anchor window. Lastly, we implement an end-of-keyword labeling method in [24], with the Δt set to 25. All baselines and our proposed method use the same network structure for feature extraction, a mini-batch of 400 utterances, and hyper-parameter tuning on the dev set. The ADAM optimizer with learning rate of 0.003 is chosen for the Deep KWS and end-of-keyword labeling systems; 0.001 is chosen for RNN-attention system.

C. Results

1) Error tradeoff comparison: In all detection error tradeoff (DET) curves figures, the test set is the 72,895 utterances of 'Hi Xiaowen', 'Nihao Wenwen' and non-keywords in the "Test" column in Table I. In computing the DET curve for 'Hi Xiaowen', both 'Nihao Wenwen' and non-keyword utterances are considered negative test data. Similarly, the DET curve for 'Nihao Wenwen' treats 'Hi Xiaowen' as negative data.

[1]It is possible that at frame t, more than one keyword is triggered. The results here count these as false hits; choosing only the maximum would lower the false hit rate.

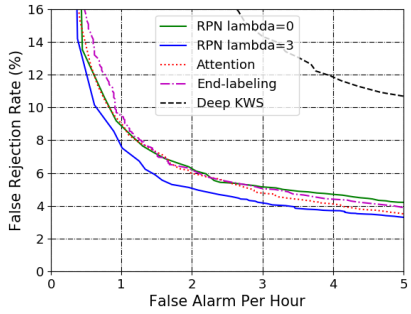

Fig. 2. DET curves on 'Hi Xiaowen'.

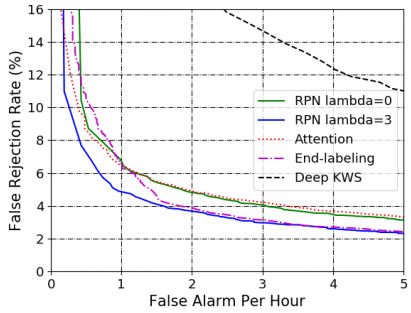

Fig. 3. DET curves on 'Nihao Wenwen'.

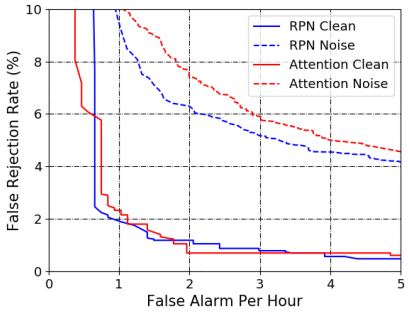

Fig. 4. RPN ($\lambda = 3$) vs. the RNN-attention system on 'Hi Xiaowen'

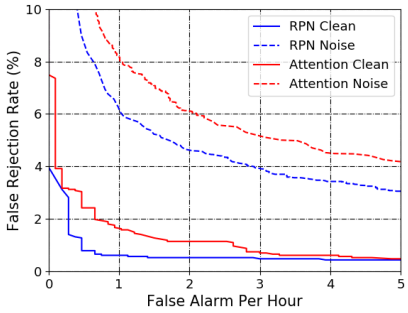

Fig. 5. RPN ($\lambda = 3$) vs. the RNN-attention system on 'Nihao Wenwen'

Both DET curves in Fig. 2 and Fig. 3 show that a non-zero λ makes RPN KWS perform much better, implying that the regression task does have a positive impact on the classification task. The finding that the DET curve for 'Nihao Wenwen' outperformed that for 'Hi Xiaowen' is likely because 'Nihao Wenwen' is longer (4 syllables instead of 3 syllables) and hence it is easier to distinguish 'Nihao Wenwen' from other non-keyword audio.

From Fig. 2 and Fig. 3, both keywords show that our method's false rejection rate (FRR) is better than all three baselines, particularly at lower false alarm (FA) rates. At 1 FA per hour, our method has achieved more than 15% relative FRR improvement on 'Hi Xiaowen', and more than 23% relative FRR improvement on 'Nihao Wenwen'. The RNN-attention method is much better than the Deep KWS baseline, consistent with previous work [22].

In Fig. 4 and 5, we show DET curves for the clean and noisy subsets separately, comparing our method to the RNN-attention baseline. Our findings are:

- Error rates are lower for clean speech, as expected;
- The benefit of the RPN approach is substantial for noisy data; and
- On clean data, there is a substantial benefit from the RPN approach for one keyword but not the other.

TABLE II
LOCATION ACCURACY: MEAN IOU AT THE SELECTED ANCHOR BEFORE AND AFTER THE RPN ('HI XIAOWEN'/'NIHAO WENWEN')

FA per hour	Pre-RPN	RPN
1	0.891/0.885	0.923/0.916
2	0.890/0.884	0.920/0.914

2) Analysis of predicted regions: We quantitatively analyzed the performance of the RPN's region proposals. Given FA=1 per hour, we calculated IoU between the ground truth box and the best region predicted by RPN, for all correctly identified positive utterances. We use the mean of IoU to evaluate the accuracy of the predicted locations.

Table II shows the location accuracy (mean IoU) for the best RPN network at two different false alarm rates for the two keywords. The Pre-RPN column gives the IoU of the best anchor before transformation, while the RPN column is after the transformation. The RPN improves accuracy, as expected. We tried thresholds for FA=1-5 per hour, and observed minimal differences in the average IoU.

IV. SUMMARY

In this paper, we propose to do keyword classification and localization jointly for KWS. Inspired by the application of RPNs in object detection, we adapt the RPN for KWS. The proposed method first selects several anchors as rough locations of the keyword in an utterance. Based on the selected anchors, the RPN will do the keyword classification for each anchor. For a positive anchor, the proposed method will transform the anchor to the ground truth region of the keyword. This method takes an end-to-end, whole-word KWS approach that directly predicts the posteriors of keywords given the anchor. To learn the keyword location, the proposed method only needs to know the starting and ending positions of the keyword (from automatic alignments), rather than subword details of the keyword. Compared with RNN-attention based systems, our method not only gets better FRR under different FA settings, it also identifies keyword locations reliably.

REFERENCES

[1] M. Saraclar and R. Sproat, "Lattice-based search for spoken utterance retrieval," in *Proc. HLT-NAACL*, 2004, pp. 129–136.

[2] D. Can and M. Saraclar, "Lattice indexing for spoken term detection," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 8, pp. 2338–2347, 2011.

[3] J. Mamou, B. Ramabhadran, and O. Siohan, "Vocabulary independent spoken term detection," in *Proc. SIGIR*, 2007, pp. 615–622.

[4] J. R. Rohlicek, W. Russell, S. Roukos, and H. Gish, "Continuous hidden Markov modeling for speaker-independent word spotting," in *Proc. ICASSP*, 1989, pp. 627–630.

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

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

[7] M.-C. Silaghi and H. Bourlard, "Iterative posterior-based keyword spotting without filler models," in *Proc. ASRU*, 1999, pp. 213–216.

[8] M.-C. Silaghi, "Spotting subsequences matching an HMM using the average observation probability criteria with application to keyword spotting," in *Proc. AAAI*, 2005, pp. 1118–1123.

[9] S. Panchapagesan, M. Sun, A. Khare, S. Matsoukas, A. Mandal, B. Hoffmeister, and S. Vitaladevuni, "Multi-task learning and weighted cross-entropy for DNN-based keyword spotting." in *Proc. INTERSPEECH*, 2016, pp. 760–764.

[10] M. Sun, D. Snyder, Y. Gao, V. Nagaraja, M. Rodehorst, N. S. Panchapagesan, S. Matsoukas, and S. Vitaladevuni, "Compressed time delay neural network for small-footprint keyword spotting," in *Proc. INTERSPEECH*, 2017, pp. 3607–3611.

[11] K. Kumatani, S. Panchapagesan, M. Wu, M. Kim, N. Strom, G. Tiwari, and A. Mandai, "Direct modeling of raw audio with DNNs for wake word detection," in *Proc. ASRU*, 2017, pp. 252–257.

[12] J. Guo, K. Kumatani, M. Sun, M. Wu, A. Raju, N. Ström, and A. Mandal, "Time-delayed bottleneck highway networks using a DFT feature for keyword spotting," in *Proc. ICASSP*, 2018, pp. 5489–5493.

[13] M. Wu, S. Panchapagesan, M. Sun, J. Gu, R. Thomas, S. N. P. Vitaladevuni, B. Hoffmeister, and A. Mandal, "Monophone-based background modeling for two-stage on-device wake word detection," in *Proc. ICASSP*, 2018, pp. 5494–5498.

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

[15] R. Prabhavalkar, R. Alvarez, C. Parada, P. Nakkiran, and T. N. Sainath, "Automatic gain control and multi-style training for robust small-footprint keyword spotting with deep neural networks," in *Proc. ICASSP*, 2015, pp. 4704–4708.

[16] T. N. Sainath and C. Parada, "Convolutional neural networks for small-footprint keyword spotting," in *Proc. INTERSPEECH*, 2015, pp. 1106–1610.

[17] J. Li, R. Zhao, Z. Chen, C. Liu, X. Xiao, G. Ye, and Y. Gong, "Developing far-field speaker system via teacher-student learning," in *Proc. ICASSP*, 2018, pp. 5699–5703.

[18] M. Woellmer, B. Schuller, and G. Rigoll, "Keyword spotting exploiting long short-term memory," *Speech Communication*, vol. 55, no. 2, pp. 252–265, 2013.

[19] Y. He, R. Prabhavalkar, K. Rao, W. Li, A. Bakhtin, and I. McGraw, "Streaming small-footprint keyword spotting using sequence-to-sequence models," in *Proc. ASRU*, 2017, pp. 474–481.

[20] S. Ö. Arık, M. Kliegl, R. Child, J. Hestness, A. Gibiansky, C. Fougner, R. Prenger, and A. Coates, "Convolutional recurrent neural networks for small-footprint keyword spotting," in *Proc. INTERSPEECH*, 2017, pp. 1606–1610.

[21] 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. SLT*, 2016, pp. 474–480.

[22] C. Shan, J. Zhang, Y. Wang, and L. Xie, "Attention-based end-to-end models for small-footprint keyword spotting," in *Proc. INTERSPEECH*, 2018, pp. 2037–2041.

[23] R. Alvarez and H. Park, "End-to-end streaming keyword spotting," in *Proc. ICASSP*, 2019, pp. 6336–6340.

[24] A. Coucke, M. Chlieh, T. Gisselbrecht, D. Leroy, M. Poumeyrol, and T. Lavril, "Efficient keyword spotting using dilated convolutions and gating," in *Proc. ICASSP*, 2019, pp. 6351–6355.

[25] H. Zhang, J. Zhang, and Y. Wang, "Sequence-to-sequence models for small-footprint keyword spotting," *arXiv preprint arXiv:1811.00348*, 2018.

[26] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with reg ion proposal networks," in *Proc. NIPS*, 2015, pp. 91–99.