TEXT ADAPTIVE DETECTION FOR CUSTOMIZABLE KEYWORD SPOTTING

Yu Xi[1], *Tian Tan*[2], *Wangyou Zhang*[1], *Baochen Yang*[1], †*Kai Yu*[1]

[1]MoE Key Lab of Artificial Intelligence, AI Institute, X-LANCE Lab, Shanghai Jiao Tong University
[2]AISpeech Ltd, Suzhou China

ABSTRACT

Always-on keyword spotting (KWS), i.e., wake word detection, has been widely used in many voice assistant applications running on smart devices. Although fixed wakeup word detection trained on specifically collected data has reached high performance, it is still challenging to build an arbitrarily customizable detection system on general found data. A deep learning classifier, similar to the one in speech recognition, can be used, but the detection performance is usually significantly degraded. In this work, we propose a novel text adaptive detection framework to directly formulate KWS as a *detection* rather than a classification problem. Here, the text prompt is used as input to promote biased classification, and a series of frame and sequence level detection criteria are employed to replace the cross-entropy criterion and directly optimize detection performance. Experiments on a keyword spotting version of Wall Street Journal (WSJ) dataset show that the text adaptive detection framework can achieve an average relative improvement of 16.88% in the detection metric F1-score compared to the baseline model.

Index Terms— streaming, wake word detection, arbitrary wake word, text prompt, training detection criteria

1. INTRODUCTION

With the development of Internet of Things (IoT), wake word detection technologies have drawn increasing interest in many real-world applications, such as voice assistants. Users can wake up their smart devices through a predefined keyword such as "OK Google", "Hey Siri", and "Alexa", and issue various control commands to devices.

The wake word detection task, which requires the model to constantly listen and decode the streaming audio, is a special case of the general keyword spotting (KWS) task. There is rich literature on the topic of keyword spotting in continuous speech processing. Offline Large Vocabulary Continuous Speech Recognition (LVCSR) systems can decode acoustic features to transcripts or lattices, which can then be used for detecting the keywords of interest [1, 2, 3, 4], but most of these systems are not suitable for low-latency wake word detection tasks in a computationally constrained environment. For online small-footprint low-latency wake word detection systems, traditional approaches are based on the keyword/filler Hidden Markov Model (HMM) [5, 6]. Alternatively, with the advance of deep learning, some prior work proposed to build systems based on a single neural network without HMM and directly predict the keyword or sub-word tokens of the keyword. The neural network in such systems include deep neural network (DNN) [7], convolutional neural network (CNN) [8, 9], recurrent neural network (RNN) or long short-term memory (LSTM) [10, 11], and the attention mechanism [12, 13]. This kind of method consists of an acoustic model and a post-processing module. The acoustic model aims to encode the speech signal into feature embeddings [14, 15] or phonetic posterior probability matrices [16, 17]. The post-processing module can be either a similarity comparison method[14] or a decoding algorithm [8, 17, 18], which aims to compute a detection score for every speech frame.

Although the aforementioned approaches can work well in some specific conditions, e.g. predefined keywords, there are still two main unsolved problems that limit their potentials. (1) Fixed keywords. Most existing wake word detection systems require predefining a specific wake word. These systems require a large amount of data that contains the predefined keywords for training, and the wake words cannot be changed after training. There has been some research [16, 19, 20, 21, 22] working on the open-vocabulary wake word detection task to allow any wake words, but there are still many remaining challenges, such as the robustness of the model, and how to detect keywords in real-time [19, 22]. (2) The mismatch between the training criterion and evaluation metrics. While the method composed of an acoustic model and a post-processing module is commonly used in the wake word detection task, most existing works adopt cross-entropy as the training criterion, which cannot well reflect the performance on evaluation metrics, such as precision, recall, false alarms, and false rejects. There are two possible directions to solve the aforementioned problem. One is to use an end-to-end model [18, 20, 22, 23, 24] to directly predict the existence probability of the keyword, and the other is to optimize detection metrics [25] at the training stage to address the mismatch between training and evaluation. In this paper, we focus on the latter direction to mitigate the mismatch and improve the performance.

In this paper, we proposed a text adaptive detection framework for the always-on keyword spotting task. We designed a streaming system with low memory consumption, which allows customizing arbitrary wake words and was optimized by detection metrics. The core contributions are summarized as following:

- We propose a wake word detection system which can **customize arbitrary wake words** using text adaptation techniques combined with a modified weighted cross-entropy loss. The loss assigns different weights to different frames according to the text adaptation information to distinguish between wake and non-wake words.
- To **address the loss-metric mismatch**, we introduce the frame-level detection metric and the sequence-level detection metric to optimize the training process directly.

2. TEXT ADAPTIVE DETECTION

In this section, we propose a novel text adaptive detection framework to directly formulate KWS as a *detection* rather than a classification problem. First, we introduce the whole framework, including the network structure, training data labeling, model input and output, etc. Later, we present how to fuse the text information with acoustic features in a streaming manner. Finally, we define the modified cross-entropy loss based on the text prompt.

†Kai Yu is the corresponding author.

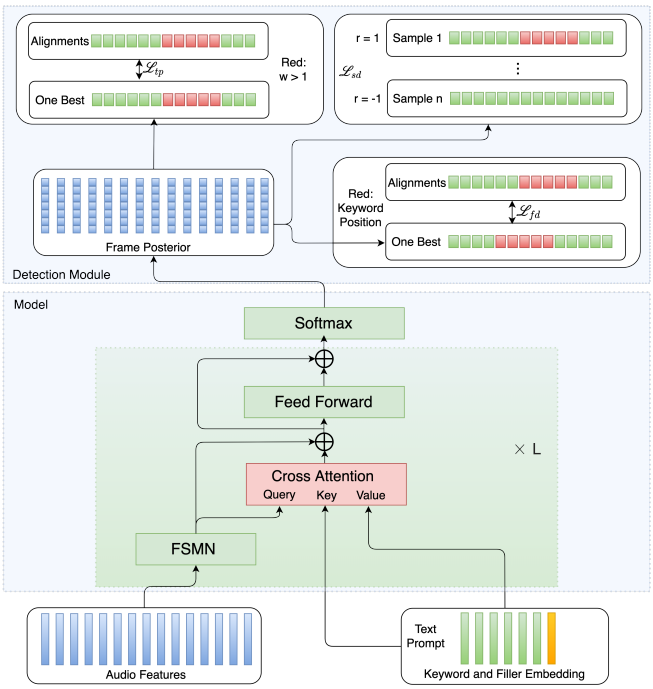

Fig. 1. The structure of the wake word detection system with the text prompt. The input to the system is the acoustic feature sequence and the modeling unit embedding corresponding to the keyword. We splice a filler embedding vector after the keyword embedding. The detection module uses the frame-level posterior matrix, time information of the keyword, and the alignments corresponding to the utterance to calculate loss and update the model.

2.1. The overview of the text adaptive detection framework

The feedforward sequential memory network (FSMN) [26] was proposed to model the long-term dependency in sequential data without using a recurrent feedback. Its modified version, named compact feedforward sequential memory network (cFSMN) [27] introduces the projection layer by combining FSMN with the low-rank matrix factorization to reduce the model size, which is favored for the small-footprint keyword spotting task. Since it has been shown superior to RNN/LSTM and can be learned more reliably and faster [26], we adopt this variant FSMN architecture as the backbone in our system to extract information from the acoustic features.

We take monophones as the modeling units. Before training the wake word detection model, we use the same dataset to pretrain a Gaussian Mixture Models-Hidden Markov Model (GMM-HMM) acoustic model for giving a forced alignment to create frame-level acoustic targets. In the training phase, the input to our model includes acoustic features, the keyword embedding sequence, and the learnable filler embedding intended to provide non-keyword text information for the non-keyword frames, like $\langle n/a \rangle$ (not applicable to any label in keyword) label in [21]. The model outputs the frame-level posterior matrix, which is used in the detection module to calculate the loss functions defined as $\mathcal{L}_{tp}$, $\mathcal{L}_{fd}$ and $\mathcal{L}_{sd}$. The overview of the proposed system is depicted in Fig. 1.

In the testing phase, the wake word detection can be done by simply feeding the test audio and the customized wake word into the well-trained model followed by a scoring algorithm [8] that includes the posterior smoothing and the confidence calculation.

2.2. The classification model with text prompt

The text prompt means to provide the text information of the keyword for the model, which is used as input to promote biased classification, making the model more sensitive to the keyword. In the training process, we randomly pick up a consecutive token sub-sequence from the mono-phone sequence of the transcript as the keyword phone sequence. Then, the acoustic embedding and token embedding are fused via the cross-attention module. During the testing phase, we can customize arbitrary keyword by using different phone sequences as the text prompt.

Some previous work [22] shared similar idea to leverage information from acoustic features and text embedding of the wake words. Our work differs from [22] in two aspects. (1) In this work, we use the text prompt to strengthen the prediction of the posterior for each frame, while in [22], text information was used to locate the position of the keyword and the detection was made for the utterance. (2) Moreover, in our work an additional filler vector is used for modeling non-wake words. All process is depict in Fig. 1.

2.3. The modified cross-entropy loss function

We use a modified cross-entropy loss function as the training criterion. The loss can be calculated as:

$$\mathcal{L}_{tp} = -\left(w \sum_{t \in W} \log p_t + \sum_{t \notin W} \log p_t \right). \quad (1)$$

Where p_t is the predicted probability of the correspond label at the t-th frame. w is a hyper-parameter used for emphasising the wake word, and W represents frames corresponding to the wake word.

The modified cross-entropy loss can be considered as a distinction between wake and non-wake words. In this work, keywords were randomly chosen for each utterance at different epoch and the start and end timestamps were obtained on the fly. Then the time information was utilized to calculate the loss defined in Equation 1. Thus, using weight $w > 1$ would naturally drive the model to pay more attention to the frames corresponding to the keyword.

3. TRAINING DETECTION MODULE

In this section, we will introduce the proposed training criteria for addressing the loss-metric mismatch in the wake word detection task. More specifically, we present the frame-level and sequence-level detection procedures, respectively, to enhance the ability to detect keywords provided by the text prompt.

3.1. Frame-level detection loss

Predicting monophone at each frame can be treated as a frame-level monophone detection task, thus, frames can be divided into four categories, i.e. true negative (TN), false positive (FP), true positive (TP), and false negative (FN). The category was determined by whether the frame belongs to the wake word or not in reference and hypothesis. For example, as shown in Fig. 2, red nodes represent the frames that belong to the keyword. If both nodes are red in Ref and Hyp, the frame is determined as a TP frame. Since the time information for Ref has been provided by the text prompt, we only need to compute the time information for Hyp. First, the hypothesis was generated by greedy search, where the phoneme with the highest probability was taken for each frame. After obtaining the one-best frame-level sequence, a de-duplication operation was applied to map the frame-level sequence to the sequence level and the index of each token was recorded. Then Knuth–Morris–Pratt (KMP) algorithm was used to find all matching points between the phone sequences generated by the text prompt and the hypothesis. It is worth noting that there may

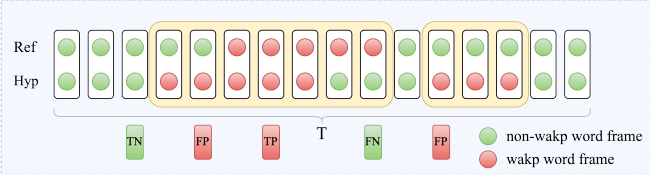

Fig. 2. Example of the frame-level detection process. In this diagram, the utterance has a total of T frames, and each frame belongs to one of the four categories: TN, FP, TP, FN. Ref and Hyp represent the reference and hypothesis, respectively. The green nodes represent the states corresponding to non-wake words, and red nodes represent the states belongs to the wake word. In Hyp sequence, there are two potential matching points, the first one is not totally aligned with reference and the latter is a false alarm.

be 0, 1 or more matching points. According to the matching result, the location of the keyword can be determined and it was then mapped back to the frame level to get the time information for the hypothesis. Finally, the frame-level detection module was used to classify each frame into one of the four classes. We demonstrate an example of two matching points in the hypothesis in Fig. 2.

The cross-entropy loss was used for optimizing frames with correct categories (TN and TP). Meanwhile, it was also used for the FN frames to push the prediction towards the correct label. However, for the FP frames, we do not care about their target categories, so we reformulated the classification of FP frames as a binary detection problem and decreased the probability to predict the wake word. The final frame-level detection loss is defined as follows:

$$\mathcal{L}_{fd} = -\Big(\sum_{t\in\{TP,TN,FN\}} \log p_t^l + \sum_{t\in\{FP\}} \log\left(1 - p_t^o\right) \Big) , \quad (2)$$

where p^l corresponds to the probability of the label category; p^o corresponds to the probability of the output prediction category.

3.2. Sequence-level detection loss

After deriving the output of the frame-level posterior matrix, we sample some frame-level sequences based on the posteriors. The sampling process is illustrated in Fig. 3. The log probability of each sequence is defined as:

$$\log P_{seq} = \sum_{t=1}^{T} \log p_t^i , \quad (3)$$

where T is the number of total frames and p_t^i is the posterior of the i-th category at the t-th time frame.

Then we utilize a de-duplication operation to merge the same consecutive monophones as in Section 3.1, thus, we get a monophone sequence for each hypothesis. Then, we can detect whether the keyword appears in the hypothetical sequence or not. The reward function was defined as: if the keyword appears, the sample gets the reward $r = 1$; otherwise, $r = -1$.

According to the rule of REINFORCE (Monte-Carlo policy gradient) [28], we can calculate the reward for each utterance as follows:

$$\mathcal{R} = \sum_{j=1}^{n} (\log P_{seq}) \cdot (r_j - baseline) , \quad (4)$$

where n is the number of samples sampled by the sampler and j represents each sample's index. The variable $baseline$ is defined as:

$$baseline = \frac{\sum_{j=1}^{n} r_j}{n} . \quad (5)$$

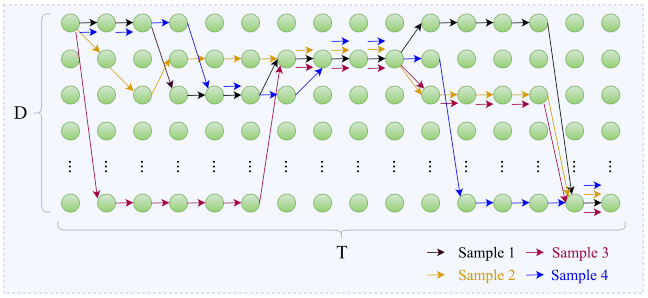

Fig. 3. Example of sampling frame-level sequences. In this diagram, the utterance has a total of T frames, and each frame can be predicted into D categories. The sequence is sampled from the categorical distribution and the total number of samples for every utterance is $n = 4$.

Finally, the sequence-level loss function for training the detection module with the policy gradient strategy is:

$$\mathcal{L}_{sd} = -\mathcal{R} . \quad (6)$$

4. EXPERIMENTS AND RESULTS

4.1. Dataset

The Wall Street Journal (WSJ) [29, 30] is used to evaluate the effectiveness of our ideas. As [17], we use scripts in Kaldi [31] for data preparation. In the training process, each word has a chance to be randomly selected as the keyword if the length of the corresponding phone sequence is between $\ell_{\min} = 3$ and $\ell_{\max} = 9$. In our experiments, $\ell_{\min}$ and $\ell_{\max}$ are determined based on the maximum and minimum keyword lengths of the test set to keep consistency between training and testing. Since our goal is to build a model to detect arbitrary wake words, we randomly select wake words in the test set based on the following rules: (1) The keyword must appear at least five times in reference to guarantee there are enough positive examples to evaluate performance and reduce randomness. (2) The keyword is not polyphonic. Since we only have a single acoustic model without any language model, we have to avoid the occurrence of polyphonic words to ensure the one-to-one mapping between words and monophone sequences. After satisfying these two requirements, we finally selected 80 words in the dev93 set, 56 words in the ev92 set, and 29 words in the ev93 set. In addition, we randomly choose 20 utterances that do not contain any of the keywords as a negative dataset. Finally, we choose 37396 utterances for training, 503 utterances for development, and 333 (ev92) and 213 (ev93) utterances for testing.

4.2. Configuration

The baseline model consists of 5 deep feedforward sequential memory network (DFSMN) [32] layers and an output layer with 75 nodes corresponding to a set of phones from the CMU pronouncing dictionary [33]. The hidden size and the projection size are 256 and 64, respectively. The larger baseline consists of 7 DFSMN layers whose hidden size is 1024 and projection size is 256. The model with the text prompt module also consists of 5 layers, which is consistent with the baseline model. Because the attention mechanism is applied to our model, we reduce the hidden size to 240 and the projection size to 48, and set the dimension of text embedding to 48, so that the total number of model parameters is comparable.

In this paper, the acoustic features are the 40-dimensional log Mel-filter bank coefficients extracted using a 25ms Hamming window with a 10ms window hop. We splice 5 frames each from the

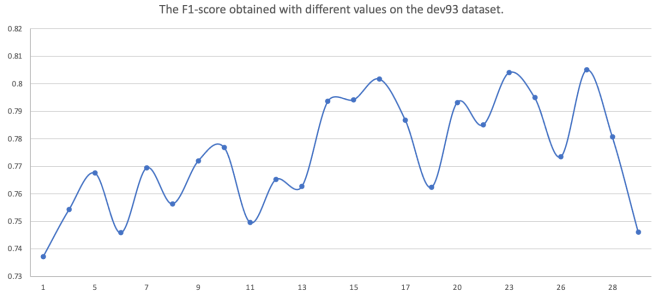

Fig. 4. The F_1-score obtained with different w values on the dev93 dataset.

left and right, for a total of 440-dimensional vectors as the input to the model. We set the parameter of frame skipping to 3 to reduce the computational cost of the model.

We use the standard cross-entropy loss to train the baseline. The proposed model with the text prompt is optimized with the $\mathcal{L}_{tp}$ in Equation (1). When the frame-level detection loss $\mathcal{L}_{fd}$ and the sequence-level detection loss $\mathcal{L}_{sd}$ proposed in Section 3 are used to optimize the model, its parameters are initialized by a well-trained model optimized by the loss $\mathcal{L}_{tp}$ with a lower learning rate. The final model is optimized with the loss $\mathcal{L}$ in a multi-task manner:

$$\mathcal{L} = \mathcal{L}_{tp} + \alpha \cdot \mathcal{L}_{fd} + \beta \cdot \mathcal{L}_{sd}\,, \tag{7}$$

as α and β are hyperparameters. We empirically set $\alpha = 1000$ and $\beta = 0.001$ as they lead to the best performance in our experiments.

4.3. Evaluation metrics

As introduced in Section 3, we incorporate the multi-level detection metrics into model training. In order to mitigate the loss-metric mismatch, we use detection statistics (TN, FP, TP, FN) to construct the loss functions. We use these statistics to compute the precision, recall, and F_1-score. For each test dataset, the micro F_1-scores are used to measure the system performance.

4.4. Impact of the weight w in loss

To study the impact of different weights w in Equation (1), we test different w values from 1 to 25 and show the results in Fig. 4. When $w = 1$, our modified loss degenerates to the original cross-entropy loss and gets the worst performance. The experimental results show that the best performance is achieved around $w = 15$. For comparison, we also finetune the model with the training detection module using the original cross-entropy with $\mathcal{L}_{fd}$ and $\mathcal{L}_{sd}$, instead of the training criterion proposed in Equation (7). The result shows this does not work well compared to use $w = 15$, so it is not presented in the table. Therefore, in the following subsection, we only present the experiment results related to $\mathcal{L}_{tp}$ that are obtained with $w = 15$.

4.5. Results

We present the experiment results [1] in Table 1. For simplicity, we only show the F1-score, reflecting the combined precision and recall results. The architecture of the baseline system is classic and widely used in academia and industry. The baseline totally has 270k parameters, while the larger model has 4000k parameters. For comparability, our proposed model has only 268k parameters, slightly smaller than the baseline model. The number of MACs for each model was calculated on a 1-sec sample. It also shows that our proposed model has comparably small computational costs as the baseline model. When just using the text prompt alone, our method can significantly outperform the baseline model and even beats the larger baseline model. It achieves a relative improvement of 14.79%, 16.77%, and 6.35% compared to the baseline model on dev93, ev92, and ev93 dataset, respectively.

[1]Detailed results for each keyword and other metrics are showed in `https://github.com/YuXI-Chn/ICASSP2022-ExpResults`

Table 1. Comparison of F_1-scores of the baselines, the attention model with the text prompt, and the attention model with the text prompt and training detection module.

Model	**#Param. (#MACs)**	**Dataset**		
		Dev93	Ev92	Ev93
Baseline	270K (8.47M)	0.692	0.762	0.844
Larger Baseline	4000K (122.82M)	0.780	0.825	0.883
Text Prompt ($\mathcal{L}_{tp}$, $w = 15$)	268K (7.64M)	0.794	0.890	0.898
+ Frame level ($\mathcal{L}_{tp}$, $\mathcal{L}_{fd}$)		0.805	0.887	0.910
+ Sequence level ($\mathcal{L}_{tp}$, $\mathcal{L}_{sd}$)		0.832	0.901	0.912
+ Frame + Sequence ($\mathcal{L}_{tp}$, $\mathcal{L}_{fd}$, $\mathcal{L}_{sd}$)		**0.850**	**0.909**	**0.917**

Furthermore, we train the model using the text prompt with both frame-level and sequence-level training detection modules. Both approaches can further improve the performance of the basic text prompt-based model, except for the frame-level detection method on the ev92 dataset. According to our analysis of the breakdown on ev92, the text prompt-based model with the frame-level training detection module increases the recall but decreases the precision, overall resulting in a lower F1-score. An increase in recall means the reduction in the number of false alarms, which is consistent with our special treatment of FP frames in Section 3.1. It indicates that the model is trained in the direction we expect. These experimental results show that our frame-level and sequence-level training methods are both effective to assist the detection model training.

We finally merged all three modules, modified the training criterion to Equation (7), and finetuned the previous well-trained model with a lower learning rate. The combination of proposed approaches achieves a final relative improvement of 22.83%, 19.22%, and 8.59% on three datasets compared to the baseline model, respectively. Compared with the larger baseline, although its parameter size is 15 times larger than our proposed system, we still get a relative improvement of 9.03%, 10.14%, 3.85%. The above observation shows the great potential of our system and illustrates that our system is well-suited for deployment on low-resource devices and meets the requirements well for the wake word detection task.

5. CONCLUSIONS

In this paper, we proposed a text adaptive detection framework for the always-on keyword spotting task, which can customize arbitrary wake words and address the mismatch between the DNN training criteria and keyword detection metrics. We introduce the text prompt with a modified version of the cross-entropy loss to distinguish the frames that belong to the wake word and non-wake words. With the proposed text prompt module, we can randomly sample wake words during the training process and then use multi-level detection criteria to assist training. Our method outperforms the baseline by a significant margin in test datasets for the comparable model size; even when the baseline model has 15 times more parameters than ours, the proposed system still achieves better performance. In future work, we will explore more fine-grained detection criteria and further improve the generalization of the model.

6. ACKNOWLEDGEMENT

This work was supported by Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102). Experiments have been carried out on the PI super-computer at Shanghai Jiao Tong University.

7. REFERENCES

[1] Siddika Parlak and Murat Saraclar, "Spoken term detection for Turkish broadcast news," in *Proc. IEEE ICASSP*. 2008, pp. 5244–5247, IEEE.

[2] Dogan Can and Murat Saraclar, "Lattice indexing for spoken term detection," *IEEE Trans. Speech Audio Process.*, vol. 19, no. 8, pp. 2338–2347, 2011.

[3] Petr Motlícek, Fabio Valente, and Igor Szöke, "Improving acoustic based keyword spotting using LVCSR lattices," in *Proc. IEEE ICASSP*. 2012, pp. 4413–4416, IEEE.

[4] Guoguo Chen, Oguz Yilmaz, Jan Trmal, Daniel Povey, and Sanjeev Khudanpur, "Using proxies for OOV keywords in the keyword search task," in *Proc. IEEE ASRU*. 2013, pp. 416–421, IEEE.

[5] Fengpei Ge and Yonghong Yan, "Deep neural network based wake-up-word speech recognition with two-stage detection," in *Proc. IEEE ICASSP*. 2017, pp. 2761–2765, IEEE.

[6] Minhua Wu, Sankaran Panchapagesan, Ming Sun, Jiacheng Gu, Ryan Thomas, Shiv Naga Prasad Vitaladevuni, Björn Hoffmeister, and Arindam Mandal, "Monophone-based background modeling for two-stage on-device wake word detection," in *Proc. IEEE ICASSP*. 2018, pp. 5494–5498, IEEE.

[7] George Tucker, Minhua Wu, Ming Sun, Sankaran Panchapagesan, Gengshen Fu, and Shiv Vitaladevuni, "Model compression applied to small-footprint keyword spotting," in *Proc. Interspeech 2016*. 2016, pp. 1878–1882, ISCA.

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

[9] Tara N. Sainath and Carolina Parada, "Convolutional neural networks for small-footprint keyword spotting," in *Proc. Interspeech 2015*. 2015, pp. 1478–1482, ISCA.

[10] Pallavi Baljekar, Jill Fain Lehman, and Rita Singh, "Online word-spotting in continuous speech with recurrent neural networks," in *Proc. IEEE SLT*. 2014, pp. 536–541, IEEE.

[11] Ming Sun, Anirudh Raju, George Tucker, Sankaran Panchapagesan, Gengshen Fu, Arindam Mandal, Spyros Matsoukas, Nikko Strom, and Shiv Vitaladevuni, "Max-pooling loss training of long short-term memory networks for small-footprint keyword spotting," in *Proc. IEEE SLT*. 2016, pp. 474–480, IEEE.

[12] Changhao Shan, Junbo Zhang, Yujun Wang, and Lei Xie, "Attention-based end-to-end models for small-footprint keyword spotting," in *Proc. Interspeech 2018*. 2018, pp. 2037–2041, ISCA.

[13] Xiong Wang, Sining Sun, Changhao Shan, Jingyong Hou, Lei Xie, Shen Li, and Xin Lei, "Adversarial examples for improving end-to-end attention-based small-footprint keyword spotting," in *Proc. IEEE ICASSP*. 2019, pp. 6366–6370, IEEE.

[14] Guoguo Chen, Carolina Parada, and Tara N. Sainath, "Query-by-example keyword spotting using long short-term memory networks," in *Proc. IEEE ICASSP*. 2015, pp. 5236–5240, IEEE.

[15] Dhananjay Ram, Lesly Miculicich Werlen, and Hervé Bourlard, "CNN based query by example spoken term detection," in *Proc. Interspeech 2018*. 2018, pp. 92–96, ISCA.

[16] Yanzhang He, Rohit Prabhavalkar, Kanishka Rao, Wei Li, Anton Bakhtin, and Ian McGraw, "Streaming small-footprint keyword spotting using sequence-to-sequence models," in *Proc. IEEE ASRU*. 2017, pp. 474–481, IEEE.

[17] Yimeng Zhuang, Xuankai Chang, Yanmin Qian, and Kai Yu, "Unrestricted vocabulary keyword spotting using LSTM-CTC," in *Proc. Interspeech 2016*. 2016, pp. 938–942, ISCA.

[18] Zeyu Zhao and Wei-Qiang Zhang, "End-to-end keyword search system based on attention mechanism and energy scorer for low resource languages," *Neural Networks*, vol. 139, pp. 326–334, 2021.

[19] Niccolò Sacchi, Alexandre Nanchen, Martin Jaggi, and Milos Cernak, "Open-vocabulary keyword spotting with audio and text embeddings," in *Proc. Interspeech 2019*. 2019, pp. 3362–3366, ISCA.

[20] Théodore Bluche and Thibault Gisselbrecht, "Predicting detection filters for small footprint open-vocabulary keyword spotting," in *Proc. Interspeech 2020*. 2020, pp. 2552–2556, ISCA.

[21] Zuozhen Liu, Ta Li, and Pengyuan Zhang, "RNN-T based open-vocabulary keyword spotting in mandarin with multi-level detection," in *Proc. IEEE ICASSP*. 2021, pp. 5649–5653, IEEE.

[22] Bo Wei, Meirong Yang, Tao Zhang, Xiao Tang, Xing Huang, Kyuhong Kim, Jaeyun Lee, Kiho Cho, and Sung-Un Park, "End-to-end transformer-based open-vocabulary keyword spotting with location-guided local attention," in *Proc. Interspeech 2021*, 2021, pp. 361–365.

[23] Kartik Audhkhasi, Andrew Rosenberg, Abhinav Sethy, Bhuvana Ramabhadran, and Brian Kingsbury, "End-to-end ASR-free keyword search from speech," *IEEE J. Sel. Top. Signal Process.*, vol. 11, no. 8, pp. 1351–1359, 2017.

[24] Niccolò Sacchi, Alexandre Nanchen, Martin Jaggi, and Milos Cernak, "Open-vocabulary keyword spotting with audio and text embeddings," in *Proc. Interspeech 2019*. 2019, pp. 3362–3366, ISCA.

[25] Ashish Shrivastava, Arnav Kundu, Chandra Dhir, Devang Naik, and Oncel Tuzel, "Optimize what matters: Training DNN-HMM keyword spotting model using end metric," in *Proc. IEEE ICASSP*. 2021, pp. 4000–4004, IEEE.

[26] Shiliang Zhang, Cong Liu, Hui Jiang, Si Wei, Lirong Dai, and Yu Hu, "Feedforward sequential memory networks: A new structure to learn long-term dependency," *arXiv preprint arXiv:1512.08301*, 2015.

[27] Shiliang Zhang, Hui Jiang, Shifu Xiong, Si Wei, and Li-Rong Dai, "Compact feedforward sequential memory networks for large vocabulary continuous speech recognition," in *Proc. Interspeech 2016*. 2016, pp. 3389–3393, ISCA.

[28] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in neural information processing systems*, 2000, pp. 1057–1063.

[29] John S. Garofolo, David Graff, Doug Paul, and Pallett David, *LDC Catalog: CSR-I (WSJ0) Complete LDC93S6A*, Philadelphia: Linguistic Data Consortium, 1993.

[30] Linguistic Data Consortium and NIST Multimodal Information Group, *LDC Catalog: CSR-II (WSJ1) Complete LDC94S13A*, Philadelphia: Linguistic Data Consortium, 1994.

[31] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely, "The Kaldi speech recognition toolkit," in *Proc. IEEE ASRU*, 2011.

[32] Shiliang Zhang, Ming Lei, Zhijie Yan, and Lirong Dai, "Deep-FSMN for large vocabulary continuous speech recognition," in *Proc. IEEE ICASSP*. 2018, pp. 5869–5873, IEEE.

[33] "The CMU pronouncing dictionary," `http://www.speech.cs.cmu.edu/cgi-bin/cmudict`, accessed 2021-10-06.