INTERSPEECH 2020
October 25–29, 2020, Shanghai, China

Predicting detection filters for small footprint open-vocabulary keyword spotting

Théodore Bluche, Thibault Gisselbrecht

Sonos Inc., Paris, France
first.last@sonos.com

Abstract

In this paper, we propose a fully-neural approach to open-vocabulary keyword spotting, that allows the users to include a customizable voice interface to their device and that does not require task-specific data. We present a keyword detection neural network weighing less than 250KB, in which the topmost layer performing keyword detection is predicted by an auxiliary network, that may be run offline to generate a detector for any keyword. We show that the proposed model outperforms acoustic keyword spotting baselines by a large margin on two tasks of detecting keywords in utterances and three tasks of detecting isolated speech commands. We also propose a method to fine-tune the model when specific training data is available for some keywords, which yields a performance similar to a standard speech command neural network while keeping the ability of the model to be applied to new keywords.

Index Terms: speech recognition, keyword spotting, neural networks

1. Introduction

The recent advances in automatic speech recognition (ASR), reaching close to human recognition performance [1], paved the way to natural language interaction in everyday life, making voice become a natural interface for the communication with objects. Such large-vocabulary ASR systems demand a lot of resources and computing power, but it has been shown [2] that the voice interface can run on device when the tasks are known, in a closed-ontology setting. For many interactions it may even be reduced to the detection of specific keywords, allowing to build systems small enough to run on microcontrollers (MCUs), which are cheap and have a low energy consumption [3].

A significant amount of tiny neural networks for keyword spotting (KWS) have been proposed in the past few years [4, 5]. These models yield very good keyword classification results and can run on MCUs. However, they require specific training data containing the keywords to be detected at inference. They lack flexibility because data should be collected and a new model should be trained every time a new keyword is added. On the other hand, traditional KWS approaches are either based on the output of an ASR system, looking for keywords in the transcript [6] or in the word [7, 8] or phone [9, 10] lattice, or based on acoustic models of phones, allowing to build models for keywords and "background" and computing likelihood ratios between the two [11]. End-to-end acoustic models predicting the character or phone sequence directly lead to efficient decoding and keyword spotting [12, 13, 14, 15] and can take into account the confusions of the network to improve the keyword models [16, 17]. These methods do not need keyword-specific training data, but still require some post-processing and a non-trivial confidence score calibration to transform the frame-wise phone scores into keyword confidences.

Recently, ASR-free approaches have been proposed, which consist in computing embeddings for both the audio and the keyword pronunciation and directly predict a keyword detection score [18, 19]. They combine the simplicity of end-to-end KWS methods and the flexibility of acoustic KWS, and do not require specific training data. In [18], the whole spoken utterance is embedded into a single vector with a recurrent auto-encoder. Similarly, the keyword is embedded into a vector using an auto-encoder of the phone sequence. The concatenation of both vectors is fed to a small neural network predicting whether the keyword appears in the utterance. In [19], different recurrent neural networks are trained to predict the word and phone embeddings. The classification is based on the distance between the keyword and utterance embedding.

In a similar vein, we propose a fully neural architecture for KWS which can be trained on generic ASR data, without specific examples of the keywords to be detected at inference. It is made of three components. An acoustic encoder, composed of a stack of recurrent layers, is pretrained as a quantized ASR acoustic model. Its intermediate features are fed to a convolutional keyword detector network trained to output keyword confidences. The weights of the latter are predicted by a keyword encoder neural network: a recurrent neural network applied to the keyword pronunciations to predict the weights of the topmost convolution kernel of the keyword detector network. This idea is similar to other works on dynamic convolution filters in computer vision for weather prediction [20], visual question answering [21], or video and stereo prediction [22].

We experimented this approach on two tasks: a continuous KWS task where keywords are detected inside queries formulated in natural language, and a speech command task where the goal is to predict one command among a predefined set. We compare this system to acoustic KWS approaches and we show that the proposed neural approach outperforms them by a large margin. We also show how the model may be fine-tuned with specific training data to get close to the performance of an end-to-end KWS classification model, without losing the ability of the model to detect new out-of-vocabulary keywords.

The remaining of the paper is divided as follows. The proposed model is described in Section 2. We report the experimental results on the two tasks in Section 3 and conclude the paper in Section 4.

2. Keyword spotting neural network

When keywords to detect are known in advance, and when training data containing those keywords are available, a neural network can be trained in an end-to-end fashion to detect them [4, 5]. In this paper, we present a method to create such a neural network for any arbitrary keyword defined post-training, which does not require training data specific to these keywords.

http://dx.doi.org/10.21437/Interspeech.2020-1186

2.1. Keyword spotting neural network

The neural network is made of a stack of unidirectional LSTM layers, followed by two convolutional layers. It has one sigmoid output for each keyword, and is trained on a generic speech dataset. The top convolutional layer of the neural network computes the probability of detecting each keyword at each timestep. For keyword k, the output sequence $\mathbf{y}_k$ is computed as follows

$$\mathbf{y}_k = y_{k,1} \dots y_{k,T} = \sigma(\theta_k * F(\mathbf{x}; \theta_F)), \tag{1}$$

where σ is the sigmoid function, $*$ is the convolution operation, $F(\mathbf{x}; \theta_F)$ represents the lower layers of the neural network with parameters θ_F applied to input $\mathbf{x}$, and θ_k is the convolution kernel corresponding to keyword k.

Since the keywords to be detected are not known during the training phase, and because the network is an open-vocabulary KWS model, the parameters θ_k of the top layer cannot be trained directly. They are predicted by an auxiliary neural network: a keyword encoder, as shown in Figure 1.

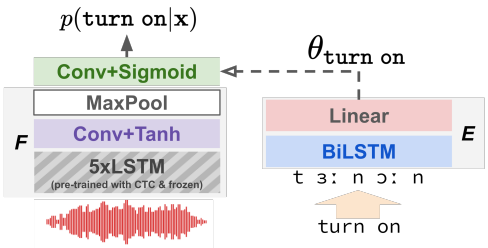


Figure 1: *Proposed model. A keyword encoder (right) predicts the weights of the top convolutional filter of the keyword spotting network (left), used to detect the keyword.*

The keyword encoder E is a neural network applied to the phone sequence π_k of a keyword k. It outputs the parameters θ_k used to detect the keyword with the KWS model:

$$\theta_k = E(\pi_k; \theta_E), \tag{2}$$

where θ_E represents the parameters of the keyword encoder network. In this work, the keyword encoder is a bidirectional LSTM network, followed by an affine transform.

2.2. Inference

At inference, the network is first configured to detect a set of n keywords $\mathcal{K} = \{k_1, \dots, k_n\}$. For each keyword $k \in \mathcal{K}$, the phone sequence π_k is retrieved from a pronunciation lexicon or a grapheme-to-phoneme converter, and convolution kernel θ_k is computed by the keyword encoder (eq. 2). Then, the top convolution layer can be created with the set of computed kernels $\{\theta_k \ ; \ k \in \mathcal{K}\}$, and the KWS network is ready to be applied to the input audio (eq. 1).

2.3. Training

The combination of eq. 1 and 2 gives:

$$\mathbf{y}_k = \sigma\left(E(\pi_k; \theta_E) * F(\mathbf{x}; \theta_F)\right). \tag{3}$$

We want $y_{k,t} \approx 1$ when the phone sequence π_k appears in the audio input $\mathbf{x}$ and ends at t, and 0 otherwise. It is therefore possible to jointly train F and E from a generic speech training set $\mathcal{D} = \{(\mathbf{x}^{(i)}, \pi^{(i)})\}$ with the cross-entropy loss, without knowing what keywords will be used at inference.

For each time step t of each training sample $\mathbf{x}^{(i)}$, we create a set $\mathcal{K}^+_{i,t}$ of positive keyword examples (i.e. $\pi_k \in \mathcal{K}^+_{i,t}$ are subsequences of $\pi^{(i)}$ ending at time t), and a set of negative keyword examples $\mathcal{K}^-_{i,t}$, and we minimize the following cross-entropy loss:

$$\mathcal{L}_{KWS} = \sum_{\mathbf{x}^{(i)}, \pi^{(i)} \in \mathcal{D}} \sum_t \ell(i,t), \tag{4}$$

where

$$\ell(i,t) = -\sum_{k \in \mathcal{K}^-_{i,t}} \log(1 - y^{(i)}_{k,t}) - \sum_{k \in \mathcal{K}^+_{i,t}} \log y^{(i)}_{k,t}. \tag{5}$$

2.4. Dataset creation

In order to build the set $\mathcal{K}^+_{i,t}$ of positive keyword examples for dataset sample i at frame t, we need to know what are the last phones appearing in the utterance at time t. They may be inferred from the forced alignment of the utterance with the ground-truth phone sequence.

To obtain them, we first train the stack of LSTMs on $\mathcal{D}$ to predict the phone sequences with the connectionist temporal classification loss (CTC [23]) $\mathcal{L}_{CTC}$:

$$\mathcal{L}_{CTC} = -\sum_{(\mathbf{x},\pi) \in \mathcal{D}} \log \sum_{l:\mathcal{B}(l)=\pi} \prod_t p(l_t|\mathbf{x}), \tag{6}$$

where $l = l_1 \dots l_T$ is a sequence of phone or *blank* labels and $\mathcal{B}$ is the CTC collapse function that removes label repetitions and *blank* labels. This network is used to align the dataset, i.e., for each $(\mathbf{x}, \pi) \in \mathcal{D}$, compute:

$$l^*(\mathbf{x}, \pi) = l^*_1 l^*_2 \dots l^*_T = \arg \max_{l:\mathcal{B}(l)=\pi} \prod_t p(l_t|\mathbf{x}). \tag{7}$$

Let W be the receptive field of the convolutional network. By removing *blanks* and label repetitions, $\mathcal{B}(l^*_{t-W} \dots l^*_t)$ will actually yield the sequence of phones in the utterance between time $t - W$ and t. Any suffix of that sequence may be considered an example of positive keyword to be detected at t. The set of positive examples $\mathcal{K}^+_t$ at time t is created by randomly sampling suffixes of $\mathcal{B}(l^*_{t-W} \dots l^*_t)$ of length 3 to 10. Note that the sampled "keyword" phone sequences do not necessarily correspond to actual words during training.

Since the network is trained with batches of dataset samples, the set of negative keyword examples $\mathcal{K}^-_t$ for one sample can merely be the union of sets of positive examples for the other samples in the batch.

2.5. Adaptation on specific dataset

Acoustic KWS approaches such as [15] rely on phone predictions and might not gain much from fine-tuning the model on a keyword-specific dataset. In the proposed approach, the weights generated by the keyword encoder could serve as a starting point for re-training on keyword-specific data, when available. That would amount to rewrite eq. 2 as:

$$\theta_k = E(\pi_k; \theta_E) + \theta_k^{(data)}. \tag{8}$$

Given a training dataset $\mathcal{D}_\mathcal{K}$ for a set of keywords $\mathcal{K}$ made of positive example of keywords and negative data, the data-specific parameters $\{\theta_k^{(data)}\}_{k \in \mathcal{K}}$ can be adjusted by gradient descent, to optimize the loss of eq. 5. By adjusting only those parameters, the ability of the model to detect any other arbitrary keyword is not lost.

3. Experimental results

3.1. Experimental setup

We trained quantized LSTM networks on Librispeech [24] with CTC and data augmentation. The inputs are MFCC and outputs are 46 phones plus a *blank* class. The details of the training and quantization procedure can be found in [15]. The weights of the LSTM cells are frozen after CTC training and not fine-tuned during the training of the KWS network, allowing to use them for ASR as well or with the method of [15]. We evaluated systems based on two such LSTM networks, with five layers of 64 and 96 units.

The top softmax layer in that model is replaced by two convolutional layers to build the KWS network (shown in purple and green in Figure 1). The first convolutional layer has a kernel of five frames, and 96 output tanh channels, followed by a max-pooling of size three and stride two. The top convolutional layer has a kernel size of 12. The total receptive field of the convolutional part has a size of 30 frames. The keyword encoder is made of a bidirectional LSTM layer with 128 units in each direction, followed by a linear transform, predicting $12 \times 96 = 1152$ weights for each keyword. Overall, the model based on the `5x64` (*resp.* `5x96`) LSTM have 208.8k (*resp.* 440.7k) parameters plus 1.2k parameters per keyword.

The keyword detector and encoder are jointly trained for five epochs using minibatches of 128 audio samples and two positive synthetic keyword samples in each $\mathcal{K}_i^+$, using the Adam optimizer and a learning rate of 1e-4, following the procedure presented in Section 2.3. All the weights, including those predicted by the keyword encoder are quantized to 8 bits.

3.2. Keyword spotting results

For this task, we crowd-sourced queries for two use-cases: a smart light scenario and a washing machine scenario[1] (cf. [15] for details). Each dataset was re-recorded in clean and noisy, reverberated far-field conditions with a SNR of 5dB. Each query contains between one and four keywords, and is expressed in natural language (e.g. "could you `[turn on]` the lights in the `[bedroom]`"). We measure the ratio of exactly parsed queries, i.e. those for which the sequence of detected keywords exactly matches the expected one, and the F1 score as a measure of performance at the keyword level.

The proposed KWS model is compared to five baselines, presented in detail in [15]. They all include the quantized LSTM acoustic model which was used as a base model for the KWS neural network. The *Viterbi* and *Lattice* baselines are LVCSR baselines, where the keywords are detected in the Viterbi decoding or in the decoding lattice of the utterance with a vocabulary of 200k and a trigram language model. The *Filler* baseline employs a standard keyword-filler model, where the filler model is a phone loop. The *Greedy* and *Sequence* baselines correspond to the approach presented in [15], similar to [12], with two post-processing methods.

We trained five models with different random initialization. The results are displayed in Figure 2 and compared to the *Sequence* post-processing approach of [15] in Table 1. The LVCSR-based results are low, although using recognition lattices provides a big improvement over Viterbi decoding. The keyword-filler model is the best of the traditional methods. The baselines from [15] are competitive with the filler model. With the same model size, they are almost always better, with both the greedy and sequence post-processors. With a smaller model, the sequence post-processor yields better results than the filler model. The proposed approach outperforms all baselines, even with the small model, which has two times less parameters.

[1] The datasets are publicly available at https://bit.ly/39YV1te

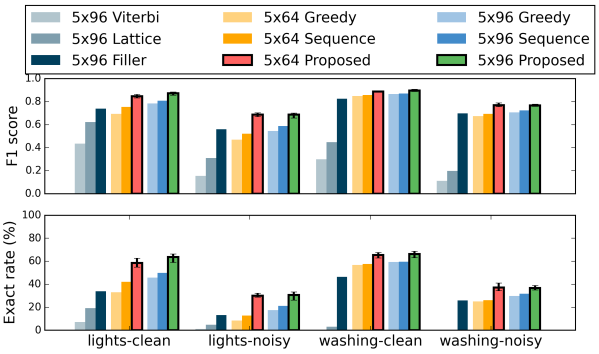


Figure 2: *F1 scores and exact rates of the proposed method (in red and green, showing the averaged results over 5 runs and the worst and best ones), compared to baselines on two keyword spotting tasks in clean and noisy conditions*

Table 1: *F1 score for the proposed model and the Sequence baseline from [15] on two keyword spotting tasks in clean and noisy conditions.*

	Dataset	lights		washing	
↓ Model	⟶	clean	noisy	clean	noisy
5x64	[15]	0.754	0.522	0.857	0.694
	Proposed	0.850	0.671	0.884	0.765
	(worst)	*0.835*	*0.654*	*0.878*	*0.751*
	(best)	*0.864*	*0.687*	*0.889*	*0.784*
5x96	[15]	0.808	0.588	0.871	0.725
	Proposed	**0.873**	**0.694**	**0.900**	**0.773**
	(worst)	*0.856*	*0.665*	*0.892*	*0.764*
	(best)	*0.883*	*0.709*	*0.911*	*0.782*

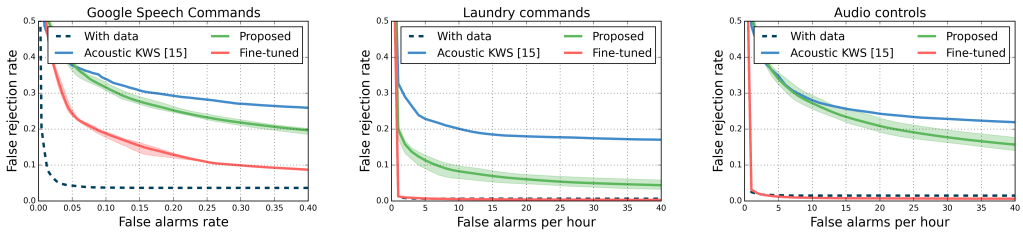
3.3. Analysis of learned filters

To analyze what the keyword encoder has learned, we compare the predicted convolution filters for different inputs. In particular, we measure the Euclidean distance between predicted filters. Indeed, if the filters are close, the KWS network will tend to make similar predictions and potentially confuse the corresponding words.

Table 2: *Keywords and closest words in a vocabulary, measured as the Euclidean distance between the predicted filters.*

Keyword	Closest vocabulary words
turn on	anon, non, turnin, fernand, maranon
decrease	crease, increase, encrease, greece
brightness	rightness, uprightness, triteness, greatness
bedroom	bathroom, begloom, broom, broome
play	flay, clay, blaye, splay, ley, lay
start	astarte, tart, stuart, upstart, sturt

The filters for all the words in a vocabulary are computed and compared to the predicted filters for some keywords. In Table 2, we show the words with the closest filters to those of the keywords. We observe that they tend to be words with similar



(a) *Google speech commands* (b) *Laundry commands* (c) *Audio controls*

Figure 3: *Speech commands results before (Section 3.4, green) and after (Section 3.5, red) fine-tuning, averaged over 5 runs, compared with a classification model fully trained on specific training data (dashed black line) and to the acoustic KWS approach of [15] (blue). For Google speech commands, the standard evaluation measuring the false alarm rate on other commands is applied.*

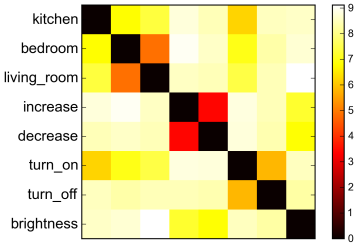


Figure 4: *Pairwise filter Euclidean distances between keywords of the lights dataset (darker is closer).*

pronunciations, of about the same length or shorter, and mostly with the same or very similar suffixes.

Figure 4 shows the pairwise distances between the filters for the keywords in the set for the *lights* dataset. We see that `increase` and `decrease` on the one hand, and `turn on` and `turn off` on the other hand are very close to one another, which correlates with the confusions of the KWS network: 37% of the confusions are between `increase` and `decrease`, 29% for `turn on` and `turn off`.

3.4. Speech commands results

For the speech command task, the goal is to detect a single command among a pre-defined set. We evaluate our approach on Google's speech command dataset [25][2] and on two in-house crowd-sourced datasets of speech commands: a dataset of audio control commands with 10 commands ("turn on, turn off, play, pause, start, stop, next track, previous track, volume up, volume down") and a dataset of laundry commands with 5 commands ("cancel, wash whites, wash delicate, wash heavy duty, wash normal"). We measure the false rejection rate across all commands on the command datasets and the number of false alarms per hour on a dataset of 45 hours of negative data made of music and speech. For the Google speech command dataset, we follow the usual evaluation procedure and measure the false alarm rate using the other commands as negative data.

We compare our results with the 5x64 model to a model trained on specific training data, similar in size (186K parameters) and performance to the `res15` ResNet architecture of [26], and to the *Greedy* acoustic KWS baseline using the same 5x64 base network [15]. The results are depicted in Figure 3. As expected, the model trained on specific training data ("with data" in black) is much better than the other two open-vocabulary approaches. Nonetheless, the proposed method outperforms the acoustic KWS baseline. The results on Google speech commands look worse, but that dataset mainly consists of short commands of two or three phonemes, which are harder to discriminate, and the negative data in this case only contains commands too.

Moreover, the proposed model is mostly similar to the one used in the acoustic KWS baseline, since only the top layers are retrained, so they could be used jointly with low computation overhead. Finally, the models labeled "with data" are models trained on specific training data while the proposed model can readily be applied to any keyword set without retraining.

3.5. Fine-tuning on keyword-specific training data

Since training data is available for these datasets, we fine-tuned the filters with the method presented in Section 2.5. The results are also shown (in red) in Figure 3. We see that after fine-tuning, the performance of the proposed model is similar to that of the model trained exclusively on specific training data. The results on Google speech commands are not as close: it might also be due to the fact that positive and negative data are short keywords in this dataset.

In these speech command tasks, the keywords are isolated: they are not inside a sentence. Before fine-tuning, the network was only trained on sentences. It is therefore possible that the gap between the network without and with fine-tuning (and "with data") is merely due to learning to detect silences surrounding the commands. This should be explored in future work. It is also worth noting that the fine-tuned network has not lost its ability to detect any arbitrary keyword, since only the weights of the top layer are modified. New keywords may then be added to the network without having to retrain it all. The baseline network does not offer these possibilities.

4. Conclusion

We presented an open-vocabulary keyword spotting system, which does not require training data specific to the keywords to be detected at inference. In contrast to most acoustic keyword spotting models, it directly predicts a confidence score at the keyword-level, alleviating the need of a confidence calibration. We have shown that the proposed model outperforms acoustic KWS baselines for the detection of keyword both inside utterances and as isolated speech commands. We proposed a method to fine-tune the model to specific training data, which makes it as good as a speech command detector trained on specific data while retaining its ability to detect other arbitrary keywords.

[2] evaluating on the same 12 commands as in [4, 26]

5. References

[1] W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, and G. Zweig, "Achieving human parity in conversational speech recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, p. 99, 2016.

[2] A. Saade, A. Coucke, A. Caulier, J. Dureau, A. Ball, T. Bluche, D. Leroy, C. Doumouro, T. Gisselbrecht, F. Caltagirone *et al.*, "Spoken language understanding on the edge," *NeurIPS Workshop on Energy Efficient Machine Learning and Cognitive Computing*, 2019.

[3] Y. Zhang, N. Suda, L. Lai, and V. Chandra, "Hello edge: Keyword spotting on microcontrollers," *CoRR*, vol. abs/1711.07128, 2017. [Online]. Available: http://arxiv.org/abs/1711.07128

[4] T. Sainath and C. Parada, "Convolutional neural networks for small-footprint keyword spotting," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[5] A. Coucke, M. Chlieh, T. Gisselbrecht, D. Leroy, M. Poumeyrol, and T. Lavril, "Efficient keyword spotting using dilated convolutions and gating," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6351–6355.

[6] J. S. Garofolo, C. G. Auzanne, and E. M. Voorhees, "The trec spoken document retrieval track: A success story." *NIST SPECIAL PUBLICATION SP*, vol. 500, no. 246, pp. 107–130, 2000.

[7] J. Mamou, D. Carmel, and R. Hoory, "Spoken document retrieval from call-center conversations," in *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, 2006, pp. 51–58.

[8] D. R. Miller, M. Kleber, C.-L. Kao, O. Kimball, T. Colthurst, S. A. Lowe, R. M. Schwartz, and H. Gish, "Rapid and accurate spoken term detection," in *Eighth Annual Conference of the international speech communication association*, 2007.

[9] M. G. Brown, J. T. Foote, G. J. Jones, K. S. Jones, and S. J. Young, "Open-vocabulary speech indexing for voice and video mail retrieval," in *Proceedings of the fourth ACM international conference on Multimedia*, 1997, pp. 307–316.

[10] Z. Chen, Y. Zhuang, and K. Yu, "Confidence measures for ctc-based phone synchronous decoding," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 4850–4854.

[11] I. Szoke, P. Schwarz, P. Matejka, L. Burget, M. Karafiát, M. Fapso, and J. Cernocky, "Comparison of keyword spotting approaches for informal continuous speech," in *Ninth European conference on speech communication and technology*, 2005.

[12] K. Hwang, M. Lee, and W. Sung, "Online keyword spotting with a character-level recurrent neural network," *arXiv preprint arXiv:1512.08903*, 2015.

[13] C. Lengerich and A. Hannun, "An end-to-end architecture for keyword spotting and voice activity detection," *arXiv preprint arXiv:1611.09405*, 2016.

[14] Z. Chen, Y. Qian, and K. Yu, "Sequence discriminative training for deep learning based acoustic keyword spotting," *Speech Communication*, vol. 102, pp. 100–111, 2018.

[15] T. Bluche, M. Primet, and T. Gisselbrecht, "Small-footprint open-vocabulary keyword spotting with quantized lstm networks," *arXiv preprint arXiv:2002.10851*, 2020.

[16] L. Lugosch, S. Myer, and V. S. Tomar, "Donut: Ctc-based query-by-example keyword spotting," *arXiv preprint arXiv:1811.10736*, 2018.

[17] Y. Yang, A. Lalitha, J. Lee, and C. Lott, "Automatic grammar augmentation for robust voice command recognition," *arXiv preprint arXiv:1811.06096*, 2018.

[18] K. Audhkhasi, A. Rosenberg, A. Sethy, B. Ramabhadran, and B. Kingsbury, "End-to-end asr-free keyword search from speech," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1351–1359, 2017.

[19] N. Sacchi, A. Nanchen, M. Jaggi, and M. Cernak, "Open-vocabulary keyword spotting with audio and text embeddings," in *INTERSPEECH 2019-IEEE International Conference on Acoustics, Speech, and Signal Processing*, no. CONF, 2019.

[20] B. Klein, L. Wolf, and Y. Afek, "A dynamic convolutional layer for short range weather prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4840–4848.

[21] H. Noh, P. Hongsuck Seo, and B. Han, "Image question answering using convolutional neural network with dynamic parameter prediction," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 30–38.

[22] X. Jia, B. De Brabandere, T. Tuytelaars, and L. V. Gool, "Dynamic filter networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 667–675.

[23] A. Graves, S. Fernandez, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 369–376.

[24] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.

[25] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *arXiv preprint arXiv:1804.03209*, 2018.

[26] R. Tang and J. Lin, "Deep residual learning for small-footprint keyword spotting," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5484–5488.