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

State sequence pooling training of acoustic models for keyword spotting

Kuba Łopatka[1], *Tobias Bocklet*[2,3]

[1]Intel Corporation
[2]Intel Labs
[3]Technische Hochschule Nürnberg, Germany

kuba.lopatka@intel.com, tobias.bocklet@th-nuernberg.de

Abstract

We propose a new training method to improve HMM-based keyword spotting. The loss function is based on a score computed with the keyword/filler model from the entire input sequence. It is equivalent to max/attention pooling but is based on prior acoustic knowledge. We also employ a multi-task learning setup by predicting both LVCSR and keyword posteriors. We compare our model to a baseline trained on frame-wise cross entropy, with and without per-class weighting. We employ a low-footprint TDNN for acoustic modeling. The proposed training yields significant and consistent improvement over the baseline in adverse noise conditions. The FRR on cafeteria noise is reduced from 13.07% to 5.28% at 9 dB SNR and from 37.44% to 6.78% at 5 dB SNR. We obtain these results with only 600 unique training keyword samples. The training method is independent of the frontend and acoustic model topology.

Index Terms: keyword spotting, machine learning, speech recognition

1. Introduction

The problem of keyword spotting (KWS) refers to detecting words of interest in an audio stream and has been an active research field during the last years. The typical use case is to trigger the digital assistant on a certain wake word. KWS usually runs locally on mobile (embedded) device, so power is a key factor [1]. This implies the need to minimize the compute and memory footprint [2], as well as to keep the false detection rate at a very low level in order to prevent unintended wakes [3].

In a typical system deep neural network acoustic models are discriminatively trained to recognize phonetic units which are known from Large Vocabulary Continuous Speech Recognition (LVCSR) [4, 5]. During decoding, frame-wise posterior scores between keyword and rejection models are combined into a final score. Keywords are represented by a sequence of phonetic units (Hidden Markov Model - HMM), while non-keywords are modelled by a filler or rejection model [6]. This architecture is referred to as DNN-HMM baseline.

Many network architectures have been tried for KWS, including affine [4], convolutional (CNN) [3, 7], time-delay (TDNN) [1] or recurrent [8, 9]. They are usually trained at frame level with cross entropy or at segment level, e.g. with Connectionist Temporal Classification (CTC) or Recurrent Neural Network Transducer (RNN-T)[10]. Some works employ per-class weighting of cross entropy to emphasize the posteriors utilized by the keyword/filler model [5, 11]. In order to solve the class imbalance problem, methods to mine regional hard examples and downsample the negative frames are introduced by Hou et al. [12] while Liu et al. employ focal loss [13].

Latest research on KWS features methods such as attention pooling [14] or max pooling [8]. The DNN directly models the probability that the keyword is spotted in the input window. In case of max pooling, the pooled score is a maximum of frame-wise scores and in case of attention pooling it is a linear combination of individual scores with weights inferred by a trainable *attention module* [15]. There are also works which feature end-to-end encoder-decoder with max-pooling loss [16, 17].

Another approach that has shown significant improvements for KWS is multi-task learning [18]. In [11] the DNN acoustic model is trained on a task of predicting the keyword-specific phone states, and an auxiliary task of predicting LVCSR senones. Sigtia et al. compare the use of LVCSR vs. keyword-specific loss in multi-task learning and show a benefit for the latter [9] while Gao et al. use a multi-condition, multi-task training approach that shows robustness with less training data [19].

In our work we try to combine the pooling and multi-task methodology. We use both LVCSR targets and keyword targets, but the main novelty is that keyword loss is computed based on the score inferred with the keyword/filler model. This is a new pooling method, which we call *state sequence pooling*.

Compared to the DNN-HMM baseline, state sequence pooling training has the following advantages:

1. It minimizes the errors in keyword detections, rather than in classifying phonetic units.
2. It emphasizes the DNN outputs used by the keyword model better than existing methods, such as per-class weighting.
3. Thanks to including a threshold in loss calculation (as outlined in Sec. 2.4) it pushes scores of positive and negative examples further apart than in baseline training.

Like max/attention pooling, our method directly models the probability of spotting the keyword in the input window. There are two major advantages of the proposed approach over known pooling methods. Firstly, state sequence pooling is based on prior phonetic knowledge, which is beneficial especially when there are few training keyword samples available. Secondly, max-pooling and attention models are typically based on recurrent topologies, which model long temporal context, spanning the whole keyword. Our method can successfully be used with a feed-forward network, which is known to be easier to train [20] or to scale down when memory is a limitation.

These advantages enable us to greatly improve the scores of the most challenging examples. We achieve a significant increase in noise robustness, which is confirmed in experiments.

2. Keyword detection algorithm

The main diagram of our algorithm is presented in Fig. 1. In the state-of-the-art training frame-wise cross entropy loss is computed by comparing acoustic model outputs with frame-level targets produced by forced alignment. We propose a new loss function which combines the frame-wise cross entropy with a

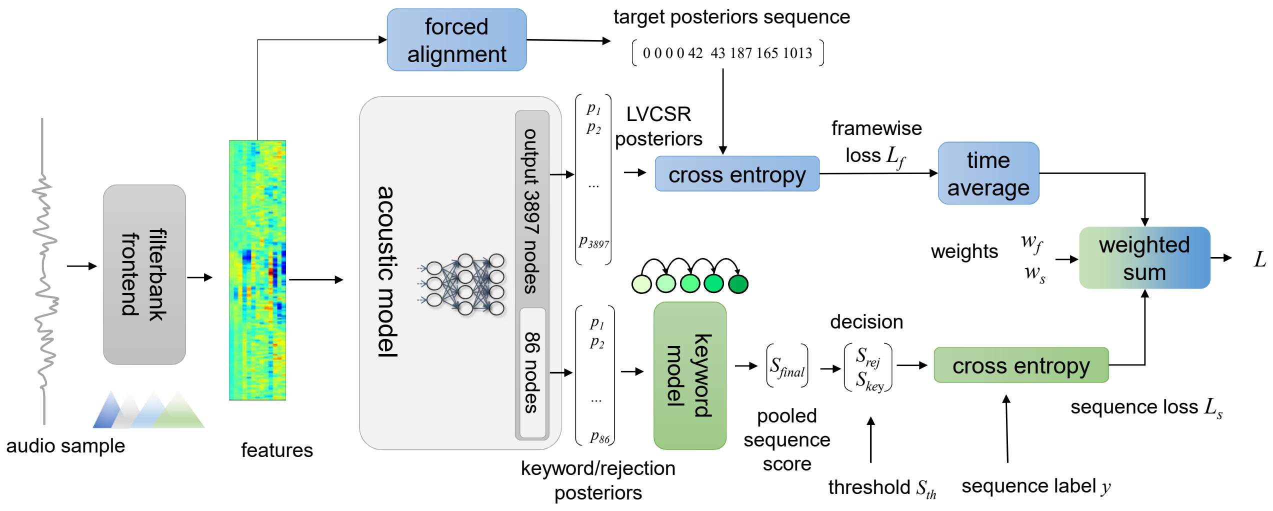

Figure 1: *Acoustic model training method (grey and blue blocks are state of the art, green blocks are introduced in this work)*

loss computed from a sequence of observations by decoding the keyword/filler model. The loss is based on the keyword score, so the acoustic model is optimized directly for the KWS task, which is an improvement over the baseline system. The setup takes advantage of multiple instance learning which is known to be beneficial for signal classification [21]. It also employs multi-task learning since it combines LVCSR and keyword loss.

A practical keyword detection system typically has two stages, where the 1st pass model works with limited resources and the 2nd pass model verifies the detection from 1st pass, with more resources available [6]. Our experiments involve a model with a low number of parameters, so it is suitable for 1st pass or for a standalone embedded detector.

2.1. Acoustic model

The acoustic model is a time-delay neural network (TDNN). Such network structure has been widely adopted for KWS due to its good tradeoff between the number of parameters and the temporal context modeled [1, 22, 23]. The topology is listed in Tab. 1. The input of the acoustic model are 40 log-filterbank features extracted from 25 ms overlapping frames with 10 ms shift. The network is prepended with a transform layer which stacks the input features and standarizes them by subtracting the mean and dividing by standard deviation. We use 3 TDNN layers with 176 units and bottleneck layers in between, all with rectified linear unit (ReLU) activation and batch normalization. The TDNN layers connect to past or future activations from previous layers, e.g. -1,0,1 means connection to 1 past, current and 1 future frame. The final layer has 3897 outputs for all posteriors which are produced during forced alignment and used as frame-level targets. Only 86 of these posteriors are used in the keyword/filler model (36 for keyword, 50 for rejection). After training the remaining outputs can be discarded, thus significantly reducing the memory footprint.

2.2. Keyword model

We use a simplified HMM decoding algorithm which is differentiable and can be used as a loss function. The keyword/rejection model is presented in Fig. 2. It comprises a sequence of states which are related to posterior probabilities of

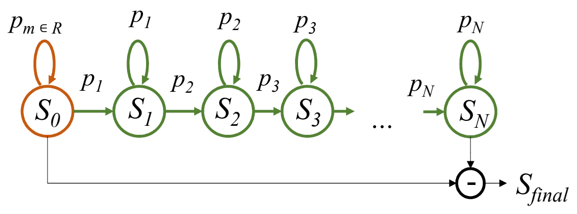

Figure 2: *State sequence model. S_0 (red) is the rejection state. States S_1 to S_N (green) are keyword states.*

phonetic units (triphone states) which constitute the phrase. The phrase used in experiments is *Hello computer*. It comprises 12 triphones, which yields $N = 36$ states (3 per triphone).

For each time step t the acoustic model estimates a vector of framewise posteriors $[p_1, p_2, p_3, ...]$. Let us assume that the posteriors are ordered in such a way, that first one relates to S_1, second to S_2 etc. The first state S_0 is a *rejection* state. It is updated based on a set R of 50 *rejection* posteriors chosen with a heuristics based on *a priori* phonetic knowledge and triphone statistics of the English language.

$$\begin{cases} S_0(0) = 0 \\ S_0(t) = S_0(t-1) + \max_{m \in R}(p_m(t)) \end{cases} \tag{1}$$

This simple approach to filler modeling ensures satisfactory performance but we find it potentially beneficial to learn the rejection posteriors data-driven within this training framework. This remains a topic for further research.

For $n = 1..N$ the state scores are updated in each time step:

$$\begin{cases} S_n(0) = -\infty \\ S_n(t) = \max\left\{S_{n-1}(t-1) + p_n(t), S_n(t-1) + p_n(t)\right\} \end{cases} \tag{2}$$

The final score of the sequence of frames is the maximum of the final state score minus the rejection state score across all time steps.

$$S_{final} = \max_{t \in \{0,1,\ldots,T-1\}} (S_N(t) - S_0(t)) \tag{3}$$

Table 1: *Topology of the acoustic model*

layer	input	nodes	parameters
shift	5*40 (-2,-1,0,1,2)	200	200
scale	200	200	200
bottleneck	200	64	12 864
TDNN	3*64 (-1,0,1)	176	33 968
bottleneck	176	64	11 328
TDNN	3*64 (-1,0,1)	176	33 968
bottleneck	176	64	11 328
TDNN	3*64 (-1,0,1)	176	33 968
affine	176	176	31 152
output	176	86	15 222
Total parameters (including batchnorm)			185 118

2.3. Baseline training

The training is composed of two steps. First, we pretrain the network for 50 iterations on a LVCSR training set without keywords to model the English phonetics properly. Next, we fine-tune for 20 more iterations on a training set including the keywords to adapt the basic system to the keywords of interest.

The *baseline model* is trained with frame-wise cross entropy loss as in Fig. 1. For comparison, we also train two models based on *weighted cross entropy*, as in [5, 11]. While computing frame-wise cross entropy we apply the weight w to the posteriors not used in the keyword/rejection model, and the weight 1 to the posteriors used in the model. We try two values of w: 0.25 and 0.5. We train our models in PyTorch with Adam optimizer and a learning rate of 0.0001.

2.4. State sequence pooling training

The final score defined in Eq. 3 is computed for each training example in the minibatch. The decision $\mathbf{d}$ is a vector whose elements relate to keywords and non-keywords (rejection):

$$\mathbf{d} = \begin{bmatrix} S_{rej} \\ S_{key} \end{bmatrix} = \begin{bmatrix} -S_{final} - (1-y) \cdot S_{th} \\ S_{final} - y \cdot S_{th} \end{bmatrix} \tag{4}$$

where y is the sequence-level label of the training example (which equals 0 for non-keywords and 1 for keywords) and S_{th} is the score threshold. Introducing the threshold term to Eq. 4 helps to shape the distribution of final keyword scores to make them more separable. In experiments we consider two values of S_{th}: 10 and 50. For example, $S_{th} = 50$ implies that the positive examples should have a score S_{final} higher than 50 and negative examples: lower than -50.

The decision $\mathbf{d}$ is then passed through softmax and compared to the label y by a cross-entropy criterion, thus obtaining a sequence-level loss L_s. The final loss is a linear combination of the sequence-level loss with the frame-wise loss L_f averaged across all frames in the sequence.

$$L = w_s \cdot L_s + w_f \cdot \frac{1}{T} \sum_{t \in \{1,\dots,T\}} L_f(t) \tag{5}$$

where w_s and w_f are the weights applied to sequence-level and frame-level loss respectively. The usage of frame-level loss enables regularization and prevents overfitting, similar as done by Povey et al. [24]. We find it best to weigh the loss components equally ($w_s = w_f = 0.5$).

The *state sequence pooling model* is trained with exactly the same data, number of epochs and hyperparameters as the baseline model.

3. Acoustic data

3.1. Training set

The training set is composed of two parts: large-vocabulary US English corpus and a keyword corpus. The English corpus cotains over 2 million utterances with a combined length of 3k hours, including Librispeech [25] and Speecon [26] databases. We augment it with 400 hours of speech with additional noise and another 400 hours with reverb + noise, so that in total we use 3.8k hours of non-keyword data.

The *keyword* corpus comprises 600 unique recordings of *Hello computer*. After augmentation with heavy noise and reverberation, 20k utterances and 80 hours of keyword data is available. We employ stratified sampling to maintain a 5:1 ratio of non-keyword vs. keyword examples in the minibatches, thus reducing the class imbalance problem.

3.2. Evaluation set

For evaluation we use 399 utterances of *Hello computer*. The test examples come from a different source and are uttered by different speakers that the ones present in the training set. Thus, we make sure that we properly test against overfitting. The evaluation is focused on additive noise as it is known from practice to be the most harmful factor. We mix the keyword recordings with the following noise types: cafeteria, music, fan and side speech. The noises used for evaluation are different signals than the ones employed for augmenting the training set. We also simulate reverberation by applying room impulse responses (RIR) simulated at 1 meter (near field) and 5 meters (far field). For non-keywords we use snippets from US English openly available podcasts. We extract 17280 random 5-second snippets which totals to 24 hours.

4. Experiments

We evaluate our models based on false rejection rate (FRR) and false accept rate (FAR). In Tab. 2 we compare the FRRs of different models for an equal number of false wakes, i.e. once in 8 hours. All models perform well on clean and room impulse response (RIR) data, without significant differences. The baseline has some deficits for noise, especially cafeteria and side speech. Weighted cross entropy helps but reducing the weight of posteriors not present in the keyword/filler model to 0.25 does not yield as good results as 0.5, which is surprising since Raju et al. used $w = 0.1$ [5]. State sequence pooling models reduce the FRR in noises by 75 % for music, by 73 % for side speech noises, and even for very difficult cafeteria 1 dB SNR - by 70 %. The score threshold $S_{th} = 10$ seems to work better for most cases than $S_{th} = 50$ but the differences are mostly insignificant at $p < 0.05$.

The plot in Fig. 3 shows the distribution of scores of keywords and non-keywords from the training set for the three models. The pretrained model yields a high separation for utterances which are clean or mixed with low levels of noise whereas difficult examples achieve low scores and result in overlapping score distributions between keywords and non-keywords. The baseline training improves the distribution by noticeably improving scores of positive examples, but still many noisy keywords yield very low scores. After training with state sequence pooling the separation is much better, precisely because the optimizer focuses on improving scores of difficult examples, i.e. samples mixed with high-level noise. In state sequence pooling training the score threshold value S_{th} is equal to 50.

Table 2: *False rejection rate (FRR [%]) at an operating point of 1 false detection in 8 hours*

condition	**Clean**	**RIR**		**cafeteria**			**music**	**fan**	**side speech**
SNR / distance		**1 m**	**5 m**	**9 dB**	**5 dB**	**1 dB**	**9 dB**	**9 dB**	**9 dB**
baseline	1.51	1.01	**0.5**	19.1	43.97	96.48	11.31	6.28	28.64
weighted cross entropy $w = 0.25$	1.76	**0.75**	0.75	15.83	42.21	95.48	12.31	4.77	24.87
weighted cross entropy $w = 0.5$	1.51	**0.75**	1.01	13.07	37.44	93.97	10.05	3.27	25.38
state sequence pooling $S_{th} = 10$	**0.5**	**0.75**	**0.5**	**5.28**	**6.78**	33.92	**2.51**	**3.02**	**7.54**
state sequence pooling $S_{th} = 50$	1.76	1.76	2.01	**5.28**	7.04	**28.64**	3.52	3.52	7.79

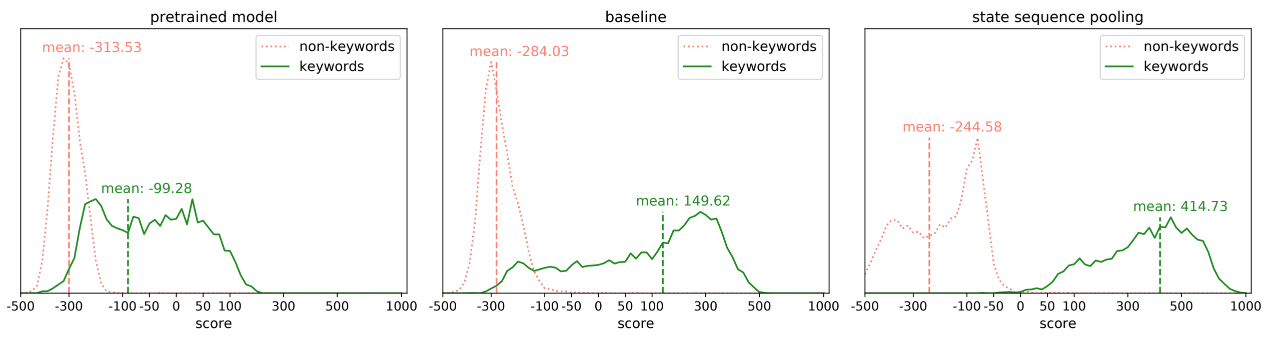

Figure 3: *Distribution of keyword and non-keyword scores after different training strategies*

Finally, in Fig. 4 we observe the detection error tradeoff for selected noise types (music 9 dB SNR and cafeteria 9 dB SNR). The state sequence pooling model (here with $S_{th} = 10$) outperforms baseline and weighted cross-entropy models by a wide margin. It enables working with a high enough threshold to only produce 1 false detection in 24 hours of continuous listening, while maintaining very high accuracy for true phrases.

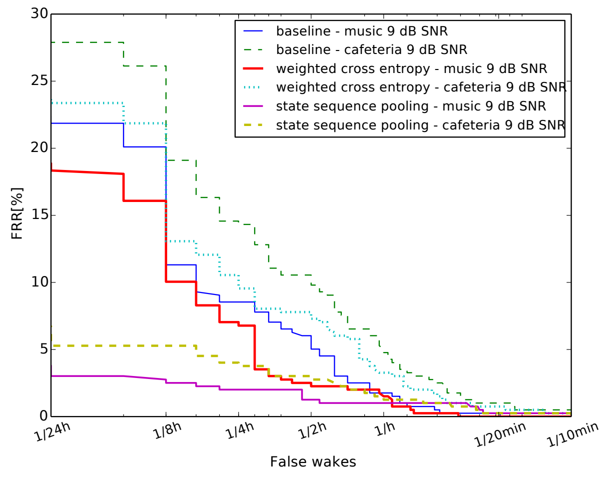

Figure 4: *Detection results in noise*

5. Conclusions

The evaluation proves that the proposed training method helps to improve KWS performance in noisy conditions. We show that the scores of noisy examples are greatly improved during training which translates to superior performance on the evaluation set as well. It is worth noting that we only use 600 unique training samples of the keyword, which are additionally augmented with noise and reverberation. Successful training with a limited set of keyword examples is one of the advantages of our method and it is thanks to using the phonetic units (senones) learnt in the LVCSR task.

In a very recent work, which was not published yet when our research started, Sigtia et al. arrived at similar conclusions [9]. We also find it that learning both LVCSR targets and keyword targets is greatly beneficial. What makes our work different is that Sigtia et al. used a BLSTM classifier trained with CTC. Such a topology is quite efficient but can only be used for 2-nd pass verification in a two-pass algorithm. Our work uses a TDNN model which can work in a 1-st pass as well. Moreover, we reuse neural network outptus across tasks, whereas Sigtia et al. reuse hidden layer parameters. We find it beneficial that the acoustic model retains the knowledge of phonetic units (senones) learnt in the LVCSR task. Finally, training with CTC requires substantially more keyword data.

There are known methods to improve KWS accuracy in noise by applying preprocessing or multichannel modeling [27, 28, 29]. We present the results on one channel without preprocessing to emphasize the improvement coming from the training setup. Still, the proposed algorithm is independent of the frontend and could be used to boost the performance of both multi-channel and one-channel systems.

The requirement of running forced alignment prior to training can be regarded as a disadvantage of our method. Even though in the presented experiments forced alignments are performed on all data, including the keyword, we believe that we could only do them for a fraction of the training set without detriment to accuracy. For those examples which do not have alignments, only state sequence loss would be backpropagated. Ideally, alignments would only be produced once for the never-changing core subset (i.e. English ASR corpora) and the additional data (i.e. task-specific keyword data) would only have sequence-level labels. This is a topic for further experiments.

6. References

[1] M. Sun *et al.*, "Compressed time delay neural network for small-footprint keyword spotting," in *Interspeech 2017*, 08 2017, pp. 3607–3611.

[2] S. Sigtia *et al.*, "Efficient voice trigger detection for low resource hardware," in *Proc. Interspeech 2018*, 2018, pp. 2092–2096. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2018-2204

[3] S. Choi *et al.*, "Temporal Convolution for Real-Time Keyword Spotting on Mobile Devices," in *Proc. Interspeech 2019*, 2019, pp. 3372–3376. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2019-1363

[4] G. Chen, C. Parada, and G. Heigold, "Small-footprint keyword spotting using deep neural networks," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 4087–4091.

[5] A. Raju *et al.*, "Data augmentation for robust keyword spotting under playback interference," *ArXiv*, vol. abs/1808.00563, 2018.

[6] M. Wu *et al.*, "Monophone-based background modeling for two-stage on-device wake word detection," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 5494–5498.

[7] T. N. Sainath and C. Parada, "Convolutional neural networks for small-footprint keyword spotting," in *INTERSPEECH*, 2015.

[8] M. Sun *et al.*, "Max-pooling loss training of long short-term memory networks for small-footprint keyword spotting," *2016 IEEE Spoken Language Technology Workshop (SLT)*, pp. 474–480, 2016.

[9] S. Sigtia *et al.*, "Multi-task learning for voice trigger detection," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 7449–7453.

[10] Y. He *et al.*, "Streaming small-footprint keyword spotting using sequence-to-sequence models," *CoRR*, vol. abs/1710.09617, 2017. [Online]. Available: http://arxiv.org/abs/1710.09617

[11] S. Panchapagesan *et al.*, "Multi-task learning and weighted cross-entropy for DNN-based keyword spotting," in *Proc. of Interspeech2016*, 09 2016, pp. 760–764.

[12] J. Hou *et al.*, "Mining effective negative training samples for keyword spotting," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7444–7448.

[13] B. Liu *et al.*, "Loss and double-edge-triggered detector for robust small-footprint keyword spotting," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 6361–6365.

[14] C. Shan, J. Zhang, Y. Wang, and L. Xie, "Attention-based end-to-end models for small-footprint keyword spotting," in *Interspeech 2018*, 09 2018, pp. 2037–2041.

[15] M. Lee *et al.*, "Orthogonality constrained multi-head attention for keyword spotting," in *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 12 2019, pp. 86–92.

[16] R. Alvarez and H. Park, "End-to-end streaming keyword spotting," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 6336–6340.

[17] H. Park, P. Violette, and N. Subrahmanya, "Learning to detect keyword parts and whole by smoothed max pooling," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 7899–7903.

[18] S. Leem, I. Yoo, and D. Yook, "Multitask learning of deep neural network-based keyword spotting for IoT devices," *IEEE Transactions on Consumer Electronics*, vol. 65, no. 2, pp. 188–194, 2019.

[19] Y. Gao *et al.*, "Towards data-efficient modeling for wake word spotting," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 7479–7483.

[20] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ser. ICML'13. JMLR.org, 2013, p. III–1310–III–1318.

[21] M. Carbonneau, V. Cheplygina, E. Granger, and G. Gagnon, "Multiple instance learning: A survey of problem characteristics and applications," *Pattern Recognition*, vol. 77, pp. 329–353, 2018.

[22] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *INTERSPEECH*, 2015.

[23] S. Myer and V. S. Tomar, "Efficient keyword spotting using time delay neural networks," *Interspeech*, vol. abs/1807.04353, 2018.

[24] D. Povey *et al.*, "Purely sequence-trained neural networks for ASR based on lattice-free MMI," 09 2016, pp. 2751–2755.

[25] 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)*, 2015, pp. 5206–5210.

[26] D. Iskra *et al.*, "SPEECON - speech databases for consumer devices: Database specification and validation," *Proceedings of LREC*, 06 2002.

[27] X. Ji *et al.*, "Integration of multi-look beamformers for multi-channel keyword spotting," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 7464–7468.

[28] Y. A. Huang, T. Hughes, T. Z. Shabestary, and T. Applebaum, "Supervised noise reduction for multichannel keyword spotting," in *ICASSP 2018 - IEEE International Conference on Acoustics, Speech and Signal Processing*, 2018, pp. 5474–5478.

[29] H. Zhang, J. Zhang, and Y. Wang, "End-to-end models with auditory attention in multi-channel keyword spotting," *CoRR*, vol. abs/1811.00350, 2018.