Building a Robust Word-Level Wakeword Verification Network

Rajath Kumar[1], *Mike Rodehorst*[1], *Joe Wang*[1], *Jiacheng Gu*[1], *Brian Kulis*[1,2]

[1]Amazon Alexa Science, Cambridge, MA
[2]Boston University, Boston, MA
{rajathku, mikerode, wangjose, jiacheg, kulibria}@amazon.com

Abstract

Wakeword detection is responsible for switching on downstream systems in a voice-activated device. To prevent a response when the wakeword is detected by mistake, a secondary network is often utilized to verify the detected wakeword. Published verification approaches are formulated based on Automatic Speech Recognition (ASR) biased towards the wakeword. This approach has several drawbacks, including high model complexity and the necessity of large vocabulary training data. To address these shortcomings, we propose to use a large receptive field (LRF) word-level wakeword model, and in particular, a convolutional-recurrent-attention (CRA) network. CRA networks use a strided small receptive field convolutional front-end followed by fixed time-step recurrent layers optimized to model the temporal phonetic dependencies within the wakeword. We experimentally show that this type of modeling helps the system to be robust to errors in the location of the wakeword as estimated by the detection network. The proposed CRA network significantly outperforms previous baselines, including an LRF whole-word convolutional network and a 2-stage DNN-HMM system. Additionally, we study the importance of pre- and post-wakeword context. Finally, the CRA network has significantly fewer model parameters and multiplies, which makes it suitable for real-world production applications.

Index Terms: wakeword spotting, voice activated devices, keyword spotting, wakeword detection, convolutional network, convolutional recurrent network, attention network

1. Introduction

In recent years, smart devices such as the Amazon Echo, Google Home, and Apple Homepod have gained immense popularity. They have become an integral part of millions of people's day-to-day lives. These smart devices include headphones, mobile phones, and speakers, etc., and are activated using a wakeword such as "Alexa" or "Ok Google". The wakeword detection[1] system is designed and modeled to be operated in noisy and challenging acoustic environments, as the devices come in various form-factors. Given the number of devices in use, unseen acoustic conditions may incorrectly cause a device to wake, yielding undesirable False Accepts (FA). These FA's initiate downstream systems in cloud services such as Automatic Speech Recognition (ASR), Natural Language Understanding (NLU), etc., and may result in unintended device behaviors. However, if we set the operating point of the wakeword detection model conservatively to prevent FA's, then we would risk many cases where real requests are ignored, i.e., false rejects (FR) degrading the user experience. Thus, a secondary *verification* network is often modeled and employed on the cloud to verify the wakeword detections.

[1]Wakeword spotting and detection are used interchangeably in the literature. In this paper, we adopt the wakeword detection terminology.

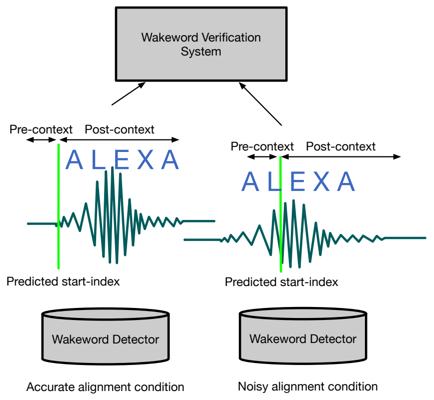

Figure 1: *System design of wakeword detectors and wakeword verification. The green line is the predicted start-index of the wakeword as inferred by the wakeword detector. Pre- and post-context is the amount of context added from the predicted start-index required by the wakeword verification system to verify the presence of the wakeword. Note the different alignments sent by different wakeword detectors affecting the context in each of the illustrated streams.*

In a real production scenario, a single wakeword verification network operates on a wide variety of devices as illustrated in Fig. 1. These devices may use a range of wakeword detection models of varying size and sophistication. Thus the verification network needs to generalize well and be robust to the addition of new devices. Conventionally, wakeword verification networks are large vocabulary ASR systems [1, 2, 3, 4]. They involve modeling the entire lexicon into the HMM, resulting in increased complexity. More recently, these ASR systems are tuned to be biased towards wakeword phonetics of interest [5, 6]. This has shown to improve wakeword verification performance significantly. Although there has been work on improving ASR performance, the ASR wakeword verification approach is of high model complexity, requires large vocabulary training data, and verification performance is heavily coupled with the word error rate (WER) objectives of the end-to-end ASR system. Most of these drawbacks are addressed by whole-word modeling when trained on a large and diverse enough number of examples. These are already widely used for wakeword detection.

In this paper, we focus on utilizing improvements in wakeword detection for the verification task. Traditionally, for low-latency wakeword detection, DNN-HMM based systems have been utilized [7, 8, 9]. These 2-stage DNN-HMM systems were further improved by better training strategies [10] and robust architectures [11]. [12] introduced a small footprint wakeword detection system that directly models the whole wakeword. This type of approach requires large amounts of wakeword-specific data for robust performance. This methodology was extended to several architectures, including convolutional networks [13], residual networks [14], temporal convolutional networks [15], and recurrent networks [16, 17, 18], which all improved significantly over [12]. More recently, self-attention layers [19] in combination with recurrent layers [20] have shown to perform better than recurrent networks for word-level modeling. Here, attention is used to re-weight the time step outputs of the recurrent layers before feeding onto feed-forward layers for classification. The receptive field of most of these networks is designed to be the average duration of the wakeword in the training corpus. We find that this design consideration degrades the performance of the model for a person with a speaking rate lower than the average. More importantly, the audio preceding and following the wakeword serves as an essential cue to indicate whether the wakeword was directed at the device in order to make a request; e.g., the wakeword spoken in the middle of a sentence is often not meant for the device. A larger receptive field is also needed for our verification model to be robust to errors in the location of the wakeword as predicted by the detector. Taking into account the learnings from wakeword detection literature and the design requirements mentioned previously, we propose using a convolutional recurrent attention architecture for the verification task.

We compare our proposed verification model against state-of-the-art wakeword detectors [13], a 2-stage DNN-HMM, confidence scores from a large vocabulary ASR model, and a large receptive field (LRF) convolutional network. The streaming wakeword detectors utilized in these experiments are similar to [13]. We refer to accurate wakeword start estimation as *accurate alignment condition* and inaccurate estimation as *noisy alignment condition*. Our experiments show that the proposed CRA network outperforms all the considered baselines in both accurate and noisy conditions. We also empirically show the importance of pre- and post-context of the wakeword. Additionally, our proposed CRA network, in comparison to the similar LRF convolutional network, operates at significantly fewer model parameters and multiplies.

2. System Design

2.1. Wakeword Detection and Input Frame

The end-to-end system comprises a wakeword detector and verification system, as shown in Fig. 1. When a wakeword is detected, N total seconds of audio context are checked by the wakeword verification model, where $N < 2$ seconds. The pre- and post- context addition described in Fig. 1 is dependent on the accuracy of the wakeword start prediction, thus may differ from the true start of the wakeword. On this audio context, 64-bin Log Filter Bank Energies (LFBE) are computed every 10 ms over a window of 25 ms resulting in an input feature frame of 195×64.

2.2. Wakeword Verification

Unlike detection, which operates in a streaming manner, wakeword verification performs a single pass on the input localized context frame. To validate our proposed CRA architecture, we consider several baseline systems that include both whole-word and phonetic based models.

2.2.1. Baselines

Phonetically modeled systems that we consider are ASR [21] and 2-stage DNN-HMM [11]. The ASR system we consider is a large vocabulary DNN-HMM system, with an n-gram language model to recognize the spoken words in the audio stream. The system outputs a confidence score for each recognized word in

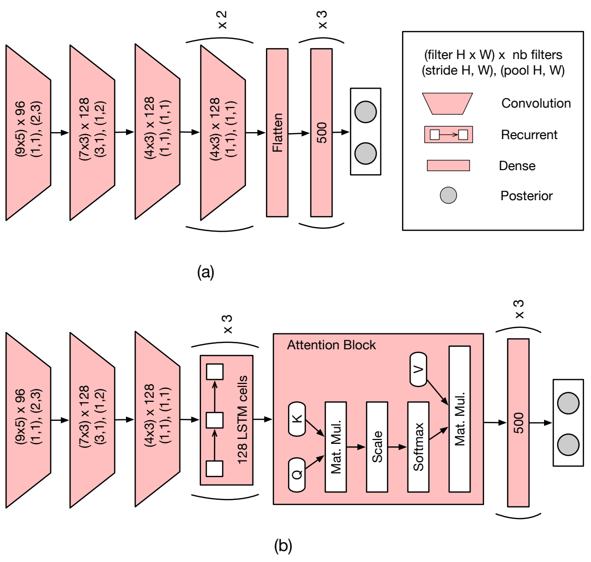

Figure 2: ***(a)*** *and* ***(b)*** *describes the architecture diagram of 195 frame receptive field convolutional network and convolutional recurrent attention system, respectively.*

the range (0,1). Similarly, for the defined wakeword, the ASR model outputs a confidence score, which we use to verify the presence of the wakeword by comparing to a fixed threshold. Since every stream is initiated with the wakeword, we set the confidence score for the wakeword to be 0 when the wakeword is not detected by ASR. Note that this ASR model is not specifically trained for the wakeword verification task, i.e., is not biased towards the wakeword. In the 2-stage DNN-HMM system, the first stage is similar to the acoustic part of the ASR system. However, unlike ASR, the 2-stage system is modeled only on the phonemes representing the wakeword and 2 additional outputs to distinguish background and silence [22]. A fixed length feature vector is extracted from the first stage DNN-HMM. This vector is engineered to aggregate information over all the frames in the hypothesized wakeword segment of the stream [22]. We pass the extracted vector as input to the second stage, which consist of multiple feed-forward layers. The second stage is a binary classification system trained to verify the presence of the wakeword. The two stages are trained separately, and the resulting score from the second stage is used for verification.

For whole-word baseline systems, we consider the detectors that were designed for streaming data. We feed in fixed-length 195-frame inputs to these detectors and consider the maximum over the scores produced as the wakeword confidence score. We consider different receptive field (RF) fully convolutional detectors (76 and 100 frames, respectively) to understand RF's effect on performance. Additionally, a convolutional network with a receptive field of 195 frames is also considered; this network need not be evaluated in a streaming fashion, as the network outputs a single score for the 195 frame input. The architecture and parameter details of the 195 RF CNN is described in Fig. 2a. The parameters of the 76 and 100 RF CNN are similar to the 195 RF CNN.

We train each of the described RF CNN's differently. As discussed earlier, the 76 and 100 RF CNN's are optimized to perform as detectors, while the 195 RF CNN is modeled for single-pass inference. The training data for the 76 RF CNN centers the wakeword in the 76 frame LFBE input. However, for the 100 RF CNN, we align the end of the wakeword towards the end of the 100 frame LFBE input. For both 76 and 100 RF CNNs, we jitter the wakeword location inside the input frame by a small margin. Our experiments show that jittering ensures model robustness during streaming inference. During the evaluation, we feed 195 frame context LFBE data to both of these CNN's on which streaming inference is performed. We use the maximum score of the output streaming posteriors for classification. For the 195 RF CNN, we provide the accurate aligned LFBE data during training as-is without adding jitter. We conducted experiments providing accurate and noisy alignment data as input during training for 195 frame RF CNN and decided on the former based on its superior verification performance in the accurate alignment condition. For all CNN's described, we report evaluations on both accurate and noisy alignment conditions.

2.2.2. Convolutional Recurrent Attention Architecture

We base the convolutional recurrent attention (CRA) architecture on the 195 RF CNN, i.e., we follow the same training and inference methodologies. Here, the convolutional front-end has a receptive field of 40 frames and is strided by 6 frames; this behaves as an efficient feature extractor to model short temporal dependencies of phonemes. The convolutional front-end, with C channels, traverses the 195 input frame, $I \in \mathbb{R}^{t \times f \times 1}$ in a streaming fashion and outputs embeddings, $D \in \mathbb{R}^{t' \times f' \times C}$. These embeddings are preserved temporally and are flattened in the frequency dimension, $D' \in \mathbb{R}^{t' \times f'C}$, i.e., all of the frequency dimensions and its respective channels at a particular time step are considered as features for that particular time. These frequency flattened embeddings, D' are then passed to the fixed timestep recurrent layers with N cells, i.e., all data during training and inference are of 195 frames and therefore the recurrent time dimension is fixed, and the recurrent states are reset at the end of 195 input frame. The output from the recurrent layers, $L \in \mathbb{R}^{t' \times N}$, are then processed by scaled dot product attention, which weights the importance of each timestep output. In the attention block, L is passed through 3 linear layers in parallel, and the outputs of each are denoted as key, K; query, Q; and value, V, respectively. The dimension of the linear layer is the same as N, i.e., $d_K = d_Q = d_V = N$. The computation inside the attention block results in the output $A \in \mathbb{R}^{t' \times N}$ as follows:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \quad (1)$$

The output A of the attention is then summed temporally, and $A' \in \mathbb{R}^N$ is obtained. A' is further passed onto feed-forward layers for classification. This architecture is illustrated in Fig. 2b, along with its parameters.

3. Results and Discussion

3.1. Experimental Procedures

We use large human-annotated anonymized datasets for training in the order of 10^6 and evaluation in the order of 10^5, where positive labels are given when a user says the wakeword to the device, and negative labels are given when the user does not say the wakeword, or says it not to the device, or the wakeword is spoken by a media source. Note that all the streams used in training for the 195 frame RF CNN and CRA architectures are from the accurate alignment condition. The evaluation datasets are grouped into accurate and noisy conditions with prior knowledge of detector models; noisy alignment streams are generally sampled from smaller and less sophisticated wakeword detection models than those used for accurate alignment streams. In practice, the accurate alignment condition has utterances where we can rely on the accuracy of the predicted start of the wakeword, whereas noisy alignment is when the estimation is not reliable. In this paper, we report evaluation for all the systems on both of these conditions.

% **decrease** in FAR in comparison to 2-stage DNN-HMM		
System	Accurate Alignment	Noisy Alignment
ASR	-6.9%	-32.1%
76_CNN	-19.4%	-41.1%
100_CNN	-34.6%	-43.9%
195_CNN	-52.6%	-31.3%
CRA	**-54.6%**	**-59.9%**

Table 1: *We report False Alarm Rate (FAR) percent decrease relative to baseline 2-stage DNN-HMM model for all considered systems and the proposed Convolutional Recurrent Attention (CRA) architecture*

We train the model for 300k steps with a batch size of 1200 samples. We use the Adam optimizer with a learning rate initialized at 0.001. We then apply an exponential moving average of 0.99 over the gradients and choose the best model across all saved checkpoints that provides the lowest average false alarm rate within a range of low miss rates on a held-out development dataset. The data in the training, development, and evaluation datasets come from disjoint sets of devices. We train all the models with a dropout of 0.3 and batch-normalization, applied after every ReLU activation in the network. For evaluating the performances of the models, we compute the percent change in False Alarm Rate (FAR) relative to a baseline at a fixed False Reject Rate (FRR). We select a low fixed FRR since instances of real requests being rejected must be low for practical use. Note that the improvements seen in FAR are not limited to the chosen FRR point.

3.2. Performance comparisons

The performance of wakeword verification systems in both accurate and noisy alignment conditions are compared in Table 1. The *100_CNN* nomenclature used in Table 1 is to be read as 100 receptive field CNN. We report false alarm rate percent decrease relative to the baseline 2-stage DNN-HMM for all considered systems at a fixed false reject rate. From Table 1, We observe in both accurate and noisy alignment condition ASR confidence scores performs the poorest with only -6.9% reduction in false alarm rate relative to 2-stage DNN-HMM in accurate alignment conditions and -32.1% in noisy alignment condition. We note that in some cases ASR does not hypothesize the wakeword and thus there is no resulting confidence score. While 2-stage DNN-HMM system performs the poorest amongst all the considered systems, ASR system performs better than 195 RF CNN in noisy alignment condition (-32.1% relative FAR reduction for ASR in comparison to -31.3% for 195 RF CNN). In all the other cases, word-level models outperform ASR system. We attribute the stronger performance of the word-level models to the availability of large training datasets. Considering various RF CNN's in accurate alignment condition, we find the performance improves as we increase the receptive field. The lowest considered receptive field, 76 RF CNN has a relative reduction in FAR of -19.4% in comparison to baseline 2-stage DNN-HMM while the highest receptive field, 195 RF CNN reduces FAR by -52.6% relative. This positive increase in FAR reduction as we increase receptive field in accurate alignment condition is not observed in noisy alignment condition. The 195 RF CNN improves over other CNNs by a

% **increase** in FAR in comparison to 50pre-145post (CRA)		
System	Accurate Alignment	Noisy Alignment
40pre-145post	+1.8%	+5.6%
30pre-145post	+3.4%	+11.4%
20pre-145post	+2.3%	+24.3%
10pre-145post	+3.4%	+65.0%
0pre-145post	+4.8%	+75.3%

Table 2: *We report performance of our convolutional recurrent attention architecture with respect to pre-context variations in accurate and noisy alignment conditions. Specifically, we compare percent increase in False Alarm Rate relative to the 195 receptive field CRA*

large margin in accurate alignment condition while performance drops significantly in noisy alignment condition (-31.3% relative FAR reduction for 195 RF CNN in comparison to -43.9% and -41.1% for 100 and 76 RF CNN respectively). We can infer from this that the local weights learned by CNN are not robust enough to generalize to wakeword position variations and is biased towards the wakeword alignment condition it was trained on i.e., accurate alignment condition. This leads to superior performance in accurate alignment condition that does not translate to noisy alignment condition. However, the smaller RF CNN's (76 and 100) that are evaluated in a streaming fashion on the same 195 frame context LFBE performs better than 195 RF CNN in noisy alignment condition, since they search for the wakeword within the available 195 LFBE frames. Comparing these systems against our CRA architecture, we find that the proposed CRA system outperforms all the models in both accurate and noisy alignment conditions (-54.6% and -59.9% relative FAR reductions in comparison to 2-stage HMM-DNN in accurate and noisy alignment conditions). The CRA and 195 RF CNN have the same receptive field, however we observe significant gains in noisy alignment condition (-59.9% relative FAR reduction for CRA in comparison to -31.3% for 195 RF CNN) while observing slightly better performance than the 195 RF CNN in the accurate alignment condition (-54.6% relative FAR reduction for CRA in comparison to -52.6% for 195 RF CNN). Note that the CRA architecture is trained on accurate alignment data, and we see performance improvements in both accurate and noisy alignment conditions, unlike 195 RF CNN, where we see gain only in accurate alignment condition when compared to other word-level systems (76 and 100 RF CNN). This is because the recurrent layers in CRA architecture learn the transition of wakeword phonetics over time provided by the convolutional front-end, thus making the network robust to different occurences of the wakeword inside the 195 frame LFBE.

We tried several variations before arriving at the described parameters for the CRA architecture. We found that utilizing GRU cells provided a slightly better performance than LSTM cells. We experimented with different variants of convolutional front-ends, specifically dilated and gated convolutional layers. These variants were not optimal in terms of performance and also had higher number of multiplies. We further experimented by adding skip connections in the recurrent layers, which did not have any effect on the model performance. Most of these models resulted in a gradient explosion at the recurrent layers leading to sub-optimal convergence. Towards the end, the described CRA architecture proved to be the best of the architectures we tried in terms of computational cost and performance. Computationally, the 195 RF CNN architecture has 232M multiplies in inference mode while the CRA network has 197M multiplies. Additionally, the CRA network has only a third of the 195 RF convolutional network's parameters, thus significantly reducing its memory footprint.

% **increase** in FAR in comparison to 50pre-145post (CRA)		
System	Accurate Alignment	Noisy Alignment
50pre-135post	+2.3%	+3.8%
50pre-125post	+3.1%	+12.9%
50pre-115post	+3.9%	+3.7%
50pre-105post	+3.7%	+6.3%
50pre-95post	+8.1%	+12.1%
50pre-85post	+14.5%	+30.5%

Table 3: *We report performance of our convolutional recurrent attention architecture with respect to post-context variations in accurate and noisy alignment conditions. Specifically, we compare percent increase in False Alarm Rate relative to the 195 receptive field CRA.*

3.3. Importance of context

In a real world scenario, the speaking rates vary between people and the model size of wakeword detectors are also different affecting the accuracy of wakeword start prediction, thus it is of importance to understand the model behaviour with changes to pre- and post- context. The correlation of pre- and post- context performance for both accurate and noisy alignment conditions is described in Table 2 and 3 respectively. In this controlled experiment, we experiment by successively increasing pre- and post- context by ten frames and report percent change in false alarm rate relative to the CRA architecture with 195 receptive field. All these experiments are trained in similar settings using CRA architecture. The *50pre-145post* nomenclature used in Table 2 and 3 is to be read as a model input of 50 frame pre-context and 145 frame post-context, where pre-context ends and post-context begins at the point when the wakeword begins according to the detector model. We find that for the accurate alignment condition, pre-context provides only small gains in performance (only $+4.8\%$ increase in false alarm rate relative to *50pre-145post* when pre-context is 0); however, in noisy alignment condition, we find that pre-context has a larger impact on performance of the model ($+75.3\%$ increase in false alarm rate relative to *50pre-145post* when pre-context is 0). From this observation, we infer that the noisy detector models often estimate the wakeword start prediction too late, thus resulting in poor performance with low pre-context. In post-context addition experiments, we find the performance worsens as post-context decreases in accurate alignment condition ($+14.5\%$ increase in false alarm rate relative to *50pre-145post* when post-context is 85 frames) and noisy conditions ($+30.5\%$ increase in false alarm rate relative to *50pre-145post* when post-context is 85 frames).

4. Conclusion

In this paper, we have described a whole word modeling approach to the wakeword verification task. Our proposed solution involves a strided convolutional front-end feeding into recurrent layers with attention. We empirically show that this type of modeling helps the network to be robust to noisy wakeword localization predicted by the wakeword detectors. We establish the efficacy of our approach by comparing against strong baseline systems that include both phonetic and word-level approaches: ASR confidence scores, 2-stage DNN-HMM and 76-100-195 frame RF CNN's. In addition, we show the importance of pre- and post- context in the wakeword verification task and that our proposed approach is computationally efficient, making it practical for production applications.

5. References

[1] P. S. Cardillo, M. Clements, and M. S. Miller, "Phonetic searching vs. lvcsr: How to find what you really want in audio archives," *International Journal of Speech Technology*, vol. 5, no. 1, pp. 9–22, 2002.

[2] R. C. Rose and D. B. Paul, "A hidden markov model based keyword recognition system," in *International conference on acoustics, speech, and signal processing*. IEEE, 1990, pp. 129–132.

[3] 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.

[4] M. Weintraub, "Lvcsr log-likelihood ratio scoring for keyword spotting," in *1995 International Conference on Acoustics, Speech, and Signal Processing*, vol. 1. IEEE, 1995, pp. 297–300.

[5] A. H. Michaely, X. Zhang, G. Simko, C. Parada, and P. Aleksic, "Keyword spotting for google assistant using contextual speech recognition," in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2017, pp. 272–278.

[6] S. Sigtia, P. Clark, R. Haynes, H. Richards, and J. Bridle, "Multi-task learning for voice trigger detection," *arXiv preprint arXiv:2001.09519*, 2020.

[7] J. Wilpon, L. Miller, and P. Modi, "Improvements and applications for key word recognition using hidden markov modeling techniques," in *[Proceedings] ICASSP 91: 1991 International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 1991, pp. 309–312.

[8] M.-C. Silaghi and H. Bourlard, "Iterative posterior-based keyword spotting without filler models," in *Proceedings of the IEEE automatic speech recognition and understanding workshop*. Citeseer, 1999, pp. 213–216.

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

[10] 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 *Interspeech*, vol. 9, 2016, pp. 760–764.

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

[12] 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)*. IEEE, 2014, pp. 4087–4091.

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

[14] 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.

[15] S. Choi, S. Seo, B. Shin, H. Byun, M. Kersner, B. Kim, D. Kim, and S. Ha, "Temporal convolution for real-time keyword spotting on mobile devices," *arXiv preprint arXiv:1904.03814*, 2019.

[16] R. Kumar, V. Yeruva, and S. Ganapathy, "On convolutional lstm modeling for joint wake-word detection and text dependent speaker verification." in *Interspeech*, 2018, pp. 1121–1125.

[17] S. O. Arik, M. Kliegl, R. Child, J. Hestness, A. Gibiansky, C. Fougner, R. Prenger, and A. Coates, "Convolutional recurrent neural networks for small-footprint keyword spotting," *arXiv preprint arXiv:1703.05390*, 2017.

[18] T. Yamamoto, R. Nishimura, M. Misaki, and N. Kitaoka, "Small-footprint magic word detection method using convolutional lstm neural network," *Proc. Interspeech 2019*, pp. 2035–2039, 2019.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[20] C. Shan, J. Zhang, Y. Wang, and L. Xie, "Attention-based end-to-end models for small-footprint keyword spotting," *arXiv preprint arXiv:1803.10916*, 2018.

[21] S. H. Krishnan Parthasarathi and N. Strom, "Lessons from building acoustic models with a million hours of speech," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 6670–6674.

[22] M. Sun, V. Nagaraja, B. Hoffmeister, and S. Vitaladevuni, "Model shrinking for embedded keyword spotting," in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, 2015, pp. 369–374.