Accurate Detection of Wake Word Start and End Using a CNN

Christin Jose[*], *Yuriy Mishchenko*[*], *Thibaud Senechal, Anish Shah, Alex Escott, Shiv Vitaladevuni*

Amazon Alexa Research, United States

{chrjse,yuriym,thibauds,anishsh,escottal,shivnaga}@amazon.com

Abstract

Small footprint embedded devices require keyword spotters (KWS) with small model size and detection latency for enabling voice assistants. Such a keyword is often referred to as *wake word* as it is used to wake up voice assistant enabled devices. Together with wake word detection, accurate estimation of wake word endpoints (start and end) is an important task of KWS. In this paper, we propose two new methods for detecting the endpoints of wake words in neural KWS that use single-stage word-level neural networks. Our results show that the new techniques give superior accuracy for detecting wake words' endpoints of up to 50 msec standard error versus human annotations, on par with the conventional Acoustic Model plus HMM forced alignment. To our knowledge, this is the first study of wake word endpoints detection methods for single-stage neural KWS.

Index Terms: keyword spotting, multi-label training, speech recognition, wake word detection, deep neural network, convolutional neural network, keyword endpoints, keyword start

1. Introduction

Keyword spotting is the task of detecting keywords of interest in a continuous audio stream. It has been an active research area in speech recognition and applications recently. With the recent increase in the popularity of voice assistants such as Alexa, Hey Google, and Siri, KWS have attracted much attention in the context of on-device wake word (WW) spotting. Accurate detection of WW endpoints in an audio stream is an important feature of KWS in WW applications. Voice assistant enabled devices only start streaming audio to the cloud when the Keyword Spotter detects a WW, and streaming must start from the WW start point.

Conventional WW detection methods use 2-stage models comprising a first stage Deep Neural Network acoustic model (AM DNN) and a Hidden Markov Model (HMM) [1, 2, 3, 4, 5]. Such a keyword spotter may also have additional classifiers after the HMM, such as an SVM, to increase the accuracy of WW detection [6]. These KWS naturally provide the endpoints of the WW via the HMM output. Specifically, during runtime, these systems perform Viterbi decoding of the WW senone sequences that produces the times of the WW's start and end senones in the input audio. However, this procedure can be computationally expensive, depending on the HMM topology, and training of such a keyword spotter is very complex.

A recent work investigated KWS based on a single-stage feed-forward DNN [7]. That DNN is trained to predict sub-keyword targets and has been shown to outperform a Keyword/Filler HMM approach. Such DNNs are also attractive for running on hardware-limited devices since the size of the model can be easily controlled to fit the devices' CPU and memory

budget by changing the number of parameters in the DNN. Convolutional Neural Networks (CNNs) have also become popular for acoustic modeling and have shown improvements over the fully connected feed-forward DNNs as KWS [8].

An important functionality of WW KWS is the ability to determine the endpoints of a WW in the audio. WW endpoints are used to decide which audio will be sent to the cloud, which helps to protect user privacy and reduces the cloud-side processing costs. The recent neural keyword spotting methods discussed above, aim at improving WW detection accuracy. However, the detection of WW endpoints becomes a greater problem with those methods. E.g., compared with the conventional AM+HMM approach, no such output as the HMM senone states is available in neural KWS. The Deep keyword spotter [7] can estimate the endpoints of the keyword in an audio based on the rise of the model's posterior corresponding to the WW sub-word labeling. However, this may not detect the endpoints of the WW very accurately.

In this paper, we consider WW spotter models designed as single-stage feed-forward CNN operating on an audio context up to a second, or word-level KWS. We introduce two methods for detecting keyword endpoints in such word-level KWS. The first method uses a second regression model trained on intermediate representations of keyword spotting CNN, in order to predict the keyword endpoints inside the input window. The second method uses a novel approach of a multi-aligned CNN model trained to detect keyword in different alignments inside the input window, such as towards the start or the end of the input window. To our knowledge, these are the first methods in the literature for keyword endpointing in word-level KWS. Likewise, the approach of multi-aligned keyword modeling is novel and may be of interest to other applications. The described methods improve standard error for keyword endpoints up to 60% compared to a constant offset algorithm. Our methods have an equivalent standard error when compared with gold standard method of AM+HMM model. These methods will allow significantly simpler model training and inference as well as better keyword spotting accuracy.

2. Word-level Keyword Spotting model

The word-level (WL) keyword spotter considered here is a CNN WW detector similar to the Deep KWS [7] and the CNN KWS [8]. However, differently from those, here the input window encompasses the entire audio context of one WW—the input window can range up to 1 sec. Specifically, the model is trained by using a set of positive examples (i.e. the audio fragments containing the WW) and negative examples (i.e. the audio fragments not containing the WW). In the WW-positive examples, the WW is consistently aligned in the long CNN input window, such as centered. The model learns to produce output representing the posterior probability of finding the WW in a given input audio fragment.

Our CNN operates on 64-dimensional Log mel Filter-Bank
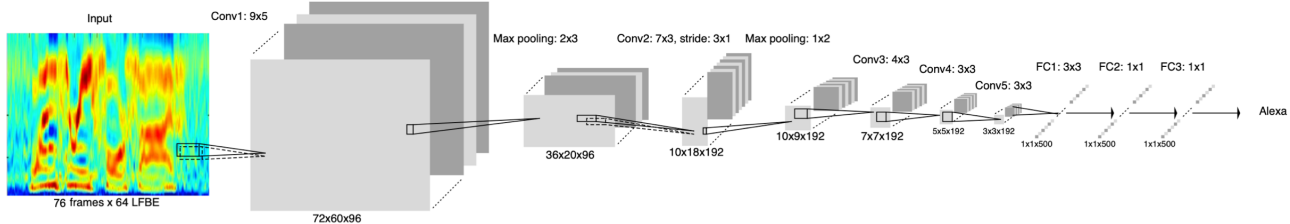
---
[*]Equal contribution

Figure 1: *Word level CNN architecture.*

Energy (LFBE) features calculated over the standard 25 msec frames with a 10 msec shift. The CNN architecture is five convolutional layers plus three fully connected (FC) layers with max-pooling after the first layer and 3-stride convolution in the second layer. See Fig. 1 for specific architecture details. Dropout and batch normalization are used with all hidden layers. The output layer is softmax over two outputs comprising "WW" and "non-WW". The model is trained using cross-entropy loss on data prepared as a mixture of WW-positive and WW-negative word-level examples, as described above, with labels in one-hot encoding. The described CNN architecture achieves superior baseline accuracy for neural WW spotting.

3. Baseline Methods for WW endpoints detection

Choosing baselines for the WW endpoints detection algorithm in the context of WL KWS is difficult as no methods for WW endpointing exist for that setting. Unlike "frame-level" models such as AM+HMM (i.e. KWS that work on sub-word input), neural WL KWS by design does not offer natural means for detecting the WW endpoints in the audio, because such models are trained to detect varying-length WW in the input window uniformly. That is, one cannot know from a detection event alone where in the input window, the WW starts and ends, without additional information about at least the WW length. We consider two baselines in this work—the AM+HMM KWS as the industry's gold standard for keyword endpointing, and a constant offset method, which is the most straightforward algorithm for WW endpointing in WL KWS.

3.1. AM+HMM endpointing

The 2-stage AM+HMM KWS [1, 2, 3, 4, 5] is de-facto the golden standard for keyword endpointing in the industry. In that algorithm, the posteriors are produced by an Acoustic Model DNN for a set of senones, based on an input audio stream, and an HMM is tuned to force-align a sequence of senones expected in the keyword to those detected. This is the classical approach used in ASR. The keyword endpoints are naturally produced in that algorithm as the times of the first and the last senones in the HMM state sequence corresponding to a keyword detection [9]. In this paper, we use a 2-stage WW model from [6] for such endpointing baseline.

3.2. Constant offset endpointing

Constant offset method is the simplest algorithm for detecting WW endpoints in WL KWS. In this approach, we depart from the view that the WW has a relatively small variation in length, such as due to pronunciation by different speakers or speed of speech differences. For example, for "Alexa" the 10-90th percentile variability in keyword length across speakers is 500 to 900 msec. In that setting, we may estimate the WW start and

end points by using a suitably chosen constant offset from the time of the WW detection event in WL KWS, given known typical WW duration and expected alignment in the keyword spotter input window. For different WWs, an optimal offset can be chosen based on a measured mean or median duration of those WWs. While simple in principle, the accuracy of this method may not be satisfactory for WWs whose length can vary greatly from the mean or median value.

4. WW endpoints detection in WL KWS

4.1. WW start-end regression model method

In this method, we add a second regression model that runs in parallel with the main detector WL CNN. The regression model uses intermediate feature representations from the hidden layers of the WL CNN as its input and is trained to output the relative offsets of the WW endpoints in the input window against ground truth, Fig 2. The ground-truth is prepared via pseudo-labels produced by the AM+HMM KWS [6]. The model is trained using the mean square error loss.
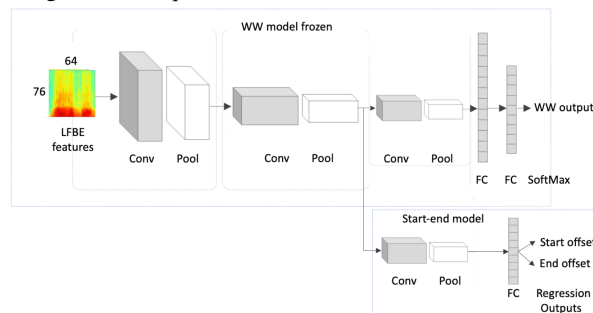

Figure 2: *Architecture of the start-end regression model.*

More specifically, we first train the main WW detector CNN using data prepared as a mixture of WW-positive and WW-negative WL examples, as described in Section 2. We then add the second branch of the start-end regression model. The key idea is that the intermediate representations from the WL CNN can allow predicting the positions of WW endpoints. We experimented with training the main detector CNN and the start-end regression model simultaneously, in a multi-task manner, or freezing the main CNN weights and training the start-end model separately. We found that freezing the main CNN was the best for WW detection accuracy.

The start-end regression model comprises one convolution layer of dimensions (5,5,200) and one fully connected layer with two outputs. We experimented with the detector CNN's hidden layer serving as the input for the start-end regression model and found the convolution layer 4 to produce the best results. The two outputs encode the start and end offsets of WW inside the input window measured in units relative to the input window length. That is, output 0 corresponds to the beginning of the input window, and output 1 corresponds to the end of the

input window. The WW can be longer than the input window. In that case, the start can be negative, and the end can be greater than 1.

4.2. Multi-aligned output WW model

In this method, we train the WL KWS with additional outputs that detect different alignments of WW inside the input window, see Fig. 3 (i). That is, one output of the CNN may be detecting WW centrally positioned inside the input window, while another output may be detecting WW positioned to start at a given frame in the input window, yet another may be detecting WW positioned to end at a given frame of the input window. The time of the peaks of each output's posteriors then allows us to detect the WW center, start, and end time points, see Fig 3 (ii).
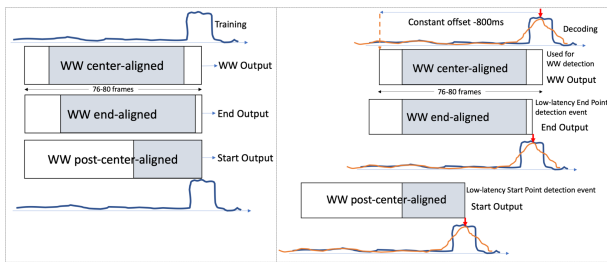

Figure 3: *Endpointing in WL KWS using multi-aligned output model. (i) Training. (ii) Inference.*

More specifically, we add two outputs to the main detector CNN output in the softmax layer, Fig. 4. The outputs are for detecting the start and end alignments of WW inside the input window. This is in addition to the main detector output, which is centrally aligned. We found such output to perform the best for the WW detection in experiments. It is also possible to use the information from all outputs together to generate WW detection events. However, we did not experiment with this option specifically.
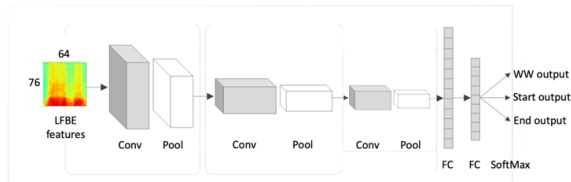

Figure 4: *Architecture of the multi-aligned WL KWS model.*

To train the model, we prepare WL training examples with WW differently aligned in the input window. An important point for start-aligned examples is that we align WW start with the middle of the input window, the "post-center-aligned" output in Fig. 3. Post-center alignment is introduced instead of a more straightforward WW alignment with the start of the input in order to reduce the latency of WW start detection. Because WL KWS has a long input window (e.g., 1 sec), the start-aligned posterior may peak significantly after the WW end for WWs that are shorter than the WL input window. We avoid this complication with post-center alignment. In this case, the WW start-aligned output becomes available before the other WW outputs. For end-aligned WW output, we prepare WL examples such that the WW end is aligned with the end of the input window up to a small margin, Fig. 3.

We train the model by mixing differently aligned WL examples in training minibatch. Specifically, we used minibatch in proportions 25%:12.5%:12.5%:50% with respect to the center, start, end aligned, and negative WW examples, respectively,

which we found to work the best in our experiments. A larger weight for the centrally aligned WW in the minibatch is given to ensure better WW detection performance. We generate the minibatch in the described manner dynamically, during training. That is, WW examples are prepared first with the context of about 2 sec and central alignment of WW inside examples. That allows selecting post-center and end-aligned WW examples later during training. During training, the WW alignment is randomly chosen during the formation of the minibatch, according to the biased die above. Random jitter is applied to the training examples by shifting the WW position slightly, to improve generalization. The examples are labeled in a one-hot manner according to their WW alignment and no-WW label.

5. Experiments and results

All models were trained for the keyword "Alexa", which is the WW for the Echo family devices at Amazon. We used 12M WW-positive and 5M WW-negative examples prepared from a corpus of annotated audio representing the far-field speech normally observed by Alexa devices in en-US locale. For training either the WW start-end regression model or WW multi-aligned output model, the WW endpoints labels are necessary. Manually annotating WW endpoints in raw audio is extremely difficult and laborious. For that reason, we used WW endpoints generated by an AM+HMM KWS [6] as pseudo-ground truth for either training and evaluation. Larger fragments of audio of 2 sec containing centered WW were prepared based on the AM+HMM KWS endpoints, as described. A set of negative examples comprising the AM+HMM KWS detections audio on data with negative annotations and central fragments of audio with negative annotations and without AM+HMM detections were added as the negative examples. WL CNN (Section 2) and Multi-aligned CNN were trained using cross-entropy loss and Adam optimizer in Tensorflow on that data for 2M steps using random initialization, mini-batch size of 4k, and learning rate of 0.001. The start-end regression model was trained for 50k steps using the frozen WL CNN model.

We tested the WW endpoints detection accuracy on held-out 33k streams. For that evaluation, we compare WW endpoints detection with pseudo-ground truth produced by the AM+HMM KWS, referred to as the "online dataset" below. Additionally, we evaluated WW start point accuracy versus human annotation on a smaller dataset. This dataset consists of 1100 "Alexa" invocations in 5 different noise conditions, including household noises, external music, pink noise, and pink noise plus music. This dataset is referred to as the "human-annotated dataset" below. We also evaluated the WW detection accuracy with respect to possible degradation because of the addition of WW endpoints detection models.

These results are presented in Table 1. The notation used in that table is as follows. The baseline *cnn_const* is the constant offset baseline described in the methods section. *cnn_align* is the multi-aligned outputs model. *cnn_regression_multi_task* is the WW start-end regression model where the main CNN detector and the start-end model were trained simultaneously. *thres_crossing* and *local_max* are the same where the main CNN detector's weights were frozen for the training of the start-end regression model. *thres_crossing* and *local_max* are the versions of *cnn_regression* differing in the way the output of the start-end regression model is used during inference on streaming audio. Specifically, *local_max* uses the start-end outputs at the time where the raw WW posterior achieves local maximum, for calculating the WW endpoints. *thres_crossing* uses the first

Table 1: *WW start end detection accuracy for different WW start-end detection models in this work.*

Detector Name	All streams		Long WW streams (length >800 ms)		WW detection
	Start Std Error	End Std Error	Start Std Error	End Std Error	(FRR at fixed FAR)
cnn_const	46 ms	107.2 ms	55.7 ms	116.7 ms	9.3%
cnn_regression_multi_task	17 ms	100.1 ms	18 ms	75.6 ms	10.23%
cnn_regression_thres_crossing	16.4 ms	95.3 ms	**17.8 ms**	73.3 ms	9.3%
cnn_regression_local_max	**16.2 ms**	55 ms	20.1 ms	113.3 ms	9.3%
cnn_align	22.4 ms	**40.9 ms**	23.7 ms	**49 ms**	9.3%

point of crossing by smooth posteriors of the detection threshold, for reduced detection latency. The metric used to quantify the accuracy of WW endpoints detection is the standard deviation of WW start or end errors (STD). This is calculated with respect to the AM+HMM endpoints in the "online dataset" and human annotations in the "human-annotated dataset". We present the WW endpoints accuracy split by WW length. This is because WW endpoints detection accuracy may be different for very long WWs. The metrics for the long WW category are calculated using "Alexa" examples with WW duration exceeding 800 msec (1.3k streams out of 33k in "online dataset"). WW duration of 800 msec is used here as the longest ten percentile for "Alexa".

We now examine the results in Table 1. For WW start, the multi-aligned outputs CNN makes a 60% improvement vs. the *cnn_const* baseline. With the *cnn_regression_local_max* model, we obtain a 65% improvement vs. *cnn_const*. In all models, the WW start accuracy with respect to the AM+HMM KWS on the "online dataset" is about 20 msec.

For WW end, multi-aligned output CNN makes a 65% improvement vs. *cnn_const* baseline and close to 55% improvement on long WWs. For *cnn_regression_local_max* model, we obtain a 49% improvement vs. *cnn_const*. In absolute terms, multi-aligned output CNN provides WW end detection accuracy close to 40 msec and start-end regression model 55-110 msec, depending on the prescription for which time-point to use for the streaming start-end model outputs.

In summary, both proposed methods show superior accuracy for detecting the WW start in WL KWS. However, for WW end detection, the multi-aligned output CNN performs significantly better. One reason for that may be the uncertainty related to using the WW start-end regression model's outputs in streaming audio. In streaming audio, the WW start-end regression model produces outputs continuously. Using different time-points for collecting the start-end outputs, which can be a smoothed posteriors' peak, raw posteriors' peak, first threshold crossing, leads to different WW endpoints prediction accuracy. We find that using the start-end model's outputs when the raw WW posterior reaches its local maximum provided the best accuracy for WW start and end detections. However, this prescription leads to the worst performance on long WWs. See *cnn_regression_local_max* in Table 1.

We now discuss the impact of adding WW endpoint models on the accuracy of main WW detection, the last column in Table 1. Metric used here is False Rejection Rate (FRR) at fixed False Acceptance Rate (FAR). In *cnn_regression_thres_cross* and *cnn_regression_local_max* detectors, since the weights of the detector CNN are frozen, the accuracy of WW detection is guaranteed to remain the same. However, if the WW start-end model is trained simultaneously with the main detector CNN, (*cnn_regression_multi_task*), we observe that WW detection accuracy can degrade by as much as relative 10%. Multi-aligned output method, *cnn_align*, achieves similar WW detection accuracy as the standard CNN WL KWS.

Finally, we evaluated the WW start detection accuracy using a smaller human-annotated dataset. Because "start" annotation is laborious, This dataset is small. However, we can see in Table 2 that the accuracy of WW start detection for all our WL keyword spotters are, in fact, identical to that of the gold-standard AM+HMM KWS, on human-annotated data. This accuracy is about 50 msec STD. This accuracy is comparable, albeit worse, with the human annotators' accuracy, which we estimated to be about 30 msec.

Table 2: *WW start detection accuracy in WL KWS and 2-stage AM+HMM model using a smaller human-annotated dataset.*

Detector name	Start Std error
AM+HMM	52 ms
cnn_align/regression	52 ms
2nd_human_annotation	31.2 ms

6. Conclusions and discussions

We present two new methods for WW endpoints detection in the WL keyword spotter, namely the second WW start-end regression model and the multi-aligned output modeling method. In the case of the WW start point detection, the accuracy for "Alexa" WW is found to be within 20 msec of the gold-standard AM+HMM keyword spotter and within 50 msec of human annotator, the latter being identical for the new methods and the industry gold-standard AM+HMM keyword spotter. For the WW end point detection, the accuracy for "Alexa" WW is found to be within 40 msec for multi-aligned output CNN and within 60 msec for the WW start-end regression model with respect to the gold-standard AM+HMM keyword spotter.

The multi-aligned output method provides overall superior accuracy for WW endpoints with a guaranteed KWS accuracy. It allows using a single CNN both for WW detection and WW start-end detection, thus, simplifying model training and inference. On the other hand, the WW start-end regression model can guarantee that the WW detection accuracy remains constant with the WW endpoints model added, because it can be trained with the main detector KWS weights frozen. In the multi-aligned output KWS, because the same feature representations are used for all WW alignments and WW detection, WW detection cannot be in principle decoupled from the endpoints detection, although we did not find that to create issues in practice.

Multi-aligned output models allow for early or low-latency detection of WW, i.e., by using the post-center aligned output. In particular, this output can produce a lower-accuracy WW detection by seeing a partial WW aligned post-center in the input window. Otherwise, the WL KWS incurs a constant latency of WW detection with respect, e.g., to the WW center point. This feature is of great interest to the cascade KWS or VAD settings. A unique advantage of multi-aligned outputs approach is also that it can be straightforwardly generalized to other applications, including ASR endpointing and WW detection using recurrent deep learning models such as LSTM. In this work, we considered a WW composed of a single keyword "Alexa". It is straightforward to extend both presented methods to phrase WWs such as "Ok Google".

7. References

[1] 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*, 2016.

[2] M. Sun, D. Snyder, Y. Gao, V. Nagaraja, M. Rodehorst, S. Panchapagesan, N. Strom, S. Matsoukas, and S. Vitaladevuni, "Compressed time delay neural network for small-footprint keyword spotting," 08 2017, pp. 3607–3611.

[3] M. Sun, A. Schwarz, M. Wu, N. Strom, S. Matsoukas, and S. Vitaladevuni, "An empirical study of cross-lingual transfer learning techniques for small-footprint keyword spotting," *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 255–260, 2017.

[4] K. Kumatani, S. Panchapagesan, M. Wu, M. Kim, N. Strom, G. Tiwari, and A. Mandal, "Direct modeling of raw audio with dnns for wake word detection," *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 252–257, 2017.

[5] J. Guo, K. Kumatani, M. Sun, M. Wu, A. Raju, N. Strom, and A. Mandal, "Time-delayed bottleneck highway networks using a dft feature for keyword spotting," 04 2018.

[6] M. Wu, S. Panchapagesan, M. Sun, J. Gu, R. Thomas, S. N. Prasad Vitaladevuni, B. Hoffmeister, and A. Mandal, "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] 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.

[8] T. N. Sainath and C. Parada, "Convolutional neural networks for small-footprint keyword spotting." in *INTERSPEECH*, 2015, pp. 1478–1482.

[9] L. D. Wilcox and M. A. Bush, "Wordspotting for voice editing and indexing," 1993, patent No. US5199077A, Filed September 1st., 1991, Issued Mar. 30th., 1993. [Online]. Available: https://patents.google.com/patent/US5199077