QUERY-BY-EXAMPLE KEYWORD SPOTTING USING LONG SHORT-TERM MEMORY NETWORKS

Guoguo Chen[*1] *Carolina Parada*[2] *Tara N. Sainath*[2]

[1] Center for Language and Speech Processing, Johns Hopkins University, Baltimore, MD
[2] Google Inc., Mountain View, CA
guoguo@jhu.edu carolinap@google.com tsainath@google.com

ABSTRACT

We present a novel approach to query-by-example keyword spotting (KWS) using a long short-term memory (LSTM) recurrent neural network-based feature extractor. In our approach, we represent each keyword using a fixed-length feature vector obtained by running the keyword audio through a word-based LSTM acoustic model. We use the activations prior to the softmax layer of the LSTM as our keyword-vector. At runtime, we detect the keyword by extracting the same feature vector from a sliding window and computing a simple similarity score between this test vector and the keyword vector. With clean speech, we achieve 86% relative false rejection rate reduction at 0.5% false alarm rate when compared to a competitive phoneme posteriorgram with dynamic time warping KWS system, while the reduction in the presence of babble noise is 67%. Our system has a small memory footprint, low computational cost, and high precision, making it suitable for on-device applications.

1. INTRODUCTION

With the growing popularity of voice control in mobile devices, the need for high performance, small footprint, and low computational cost keyword spotting (KWS) methods is becoming increasingly important [1]. In such applications, KWS usually serves as a frontier of voice search: it listens to the audio continuously and initiates the voice search if a specific keyword is detected, thus providing a fully hands-free experience when interacting with devices.

A common use is to have a pre-defined keyword to activate devices. For example, Google's voice search [2] uses the phrase "Okay Google" to initiate the search interface and Apple's conversational assistant Siri features the keyword sequence "Hey Siri". However, this general phrase makes the experience less personal, and usually requires additional speaker identification if the user does not want others to easily activate their device. In this work we seek a keyword spotting method that allows users to define their own keyword; for example, a user may select a keyword by saying the word or phrase a few times during enrollment. After enrollment, the user-specified keyword can be used to activate the device.

We are interested in small memory footprint and low computational cost solutions, suitable for on-device applications. While KWS is an active research area, most techniques are not suitable with our constraints. A common KWS approach is the Keyword/Filler Hidden Markov Model (HMM) [3, 4, 5, 6, 7]. It first builds a special decoding graph that contains both keywords and filler words, and then uses Viterbi decoding to determine the best path through the graph. This requires prior knowledge of the keywords, which is not appropriate for our problem. Another area of research relies on using large vocabulary continuous speech recognition (LVCSR) systems to decode the audio into lattices or confusion networks, and search the keywords from there [8, 9, 10]. The keywords, however, are only limited to the words that are already defined in the LVCSR vocabulary. While effort has been made to make such systems keyword-independent [11, 12, 13, 14], there is usually a performance degradation when the keywords are out of vocabulary. In addition, these approaches are relatively expensive because of the LVCSR system.

Given our goal of detecting user-specified keywords, query-by-example (QbyE) is the most appropriate KWS technique. QbyE methods usually take several examples of the keywords as templates, and compare the test audio segment against the templates to make detection decisions. In [15] example keywords are decoded with an LVCSR system to get their lattice representation as templates. This is computationally expensive since the LVCSR system involves speaker adaptation, discriminative features and model transformations [16]. In [17] graph-based method is proposed to embed audio segments into a fixed-dimensional space, but dynamic time warping (DTW) is performed between the test audio segment and all the training segments in order to compute the embedding, which can be slow given large number of training segments. In [18, 19], Gaussian or phoneme posteriorgrams are generated as templates from example keywords, and DTW is used to compare the templates. Though this type of DTW-based methods have well-known inadequacies [17], it is the most appropriate KWS baseline for our application.

Given the constraints of our problem and the limitations of previous QbyE approaches, we propose a novel LSTM-based feature extractor. In our approach, first an LSTM is trained with whole word output targets. Next, for each audio segment, a fixed-length representation of the audio is created by taking the activations from the last hidden layer of the LSTM and stacking them over a fixed number of frames. This embeds audio segments of different length into a fixed-dimensional space, therefore vector distance can be used for similarity measurement. Our method only requires a forward pass computation of the neural network, followed by a vector distance computation, and therefore is more efficient than [15] where an LVCSR is involved and [17] where multiple DTW computations are necessary. It also requires less computation than [18, 19] since vector distance is used instead of DTW.

To understand the behavior of our proposed LSTM KWS system, we first conduct experiments on a clean enrollment and evaluation set. We find that the LSTM KWS system reduces the false rejection rate by 86% relatively at 0.5% false alarm rate, when compared to the DTW KWS system. Next, we explore its behavior at the presence of noise. We add 10 dB babble noise to the evaluation set, and achieve 67% relative false rejection rate reduction at the same

[*] The author performed the work as a summer intern at Google, MTV.

false alarm rate. We further add 10 dB cafe noise to the enrollment set, and the reduction is 37%. Our proposed system is consistently better than the standard DTW KWS system in various environments.

2. LSTM FEATURE EXTRACTOR SYSTEM

The general idea of our proposed system is to embed audio segments of varying lengths into a fixed-dimensional representation. Given the success of deep learning [20], and the power of LSTMs for sequence modeling [21], we choose an LSTM to learn this emdedding. The LSTM is attractive because the state of the LSTM can encode information about past history, and intuitively after processing a complete audio segment, this LSTM state encodes information about the complete sequence. This idea was motivated by face verification work in [22], with the key difference in our work being that we use the LSTM state to embed a fixed-length representation of a variable length input sequence, as opposed to having a fixed-length input representation and thus using a convolutional neural network.

2.1. Feature Extraction

To reduce computation, we use a voice-activity detection system and only run the KWS algorithm in voice regions. The voice-activity detector, described in [23], uses 13-dimensional PLP features along with their deltas and double-deltas as input to a 30-component diagonal covariance GMM-trained system, which generates speech and non-speech posteriors at every frame. This is followed by a hand-tuned state machine, which performs temporal smoothing by identifying regions where speech posteriors exceed a threshold.

For the speech regions, we generate 40-dimensional log-filterbank energies computed every 10 ms over a window of 25 ms. For the deep neural network (DNN)-based system, contiguous frames are stacked to add sufficient left and right context. The input window is asymmetric since each future frame adds 10 ms of latency to the system. We use 5 future frames and 10 past frames, to provide the best trade-off between accuracy, latency, and computation [24]. For the LSTM-based system, no frames are stacked.

2.2. Long Short-Term Memory

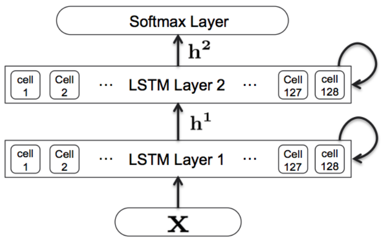

Fig. 1. LSTM architecture

Long short-term memory (LSTM) is a type of recurrent neural network (RNN) used to model long-range dependencies [21]. RNNs are used for a variety of sequence-labeling tasks. Specifically, given an input sequence $\mathbf{x} = \{x_1, \ldots, x_T\}$, an RNN computes a sequence of hidden vectors $\mathbf{h} = \{h_1, \ldots, h_T\}$ and outputs $\mathbf{y} = \{y_1, \ldots, y_T\}$ for all time steps $\{1, \ldots, T\}$. An LSTM uses special memory cells to model the temporal sequence, which allows it to more effectively exploit long-range context than an RNN.

In our work, we train a 2-layer LSTM, as shown in Figure 1. The network has 15k output targets, representing whole word units. We exclude the evaluation keywords from the 15k output targets to make the feature extractor independent of the test keywords.

2.3. LSTM Feature Extractor

After training the LSTM, given an audio segment, we use the LSTM to determine a fixed-dimensional feature vector to represent the audio signal. Our fixed length representation is created by removing the softmax layer and using the hidden units from the 2^{nd} LSTM layer. More specifically, given an acoustic feature $\mathbf{x}$ with T frames, the hidden units from the second layer of the LSTM are given as $\mathbf{h^2} = \{h_1^2, \ldots, h_T^2\}$. Here $h_i^2 \in \Re^n$, where n represents the number of LSTM cells. As each $h_i^2 \in \mathbf{h^2}$ encodes information up to time i, there is no need to use all state vectors $\mathbf{h^2}$. We create a fixed-length representation $\mathbf{f}$ by choosing the last k state vectors, as denoted by

$$\mathbf{f} = \{h_{T-k+1}^2, \ldots, h_T^2\} \quad (1)$$

The parameter k can be estimated from the enrollment templates. In our experiments, we choose k to be the averaged number of frames of all the templates as we want to encode as much information as possible. Zeros are padded in front of $\mathbf{f}$ if the segment length T is smaller than the desired template length k.

2.4. LSTM KWS

Now that we have described how to create a fixed-length representation from an audio segment, in this section we describe the enrollment and verification phases of our LSTM KWS system, as illustrated in Figure 2. In the enrollment phase, an utterance is spoken three times. For each utterance, the activations from the last hidden layer of the LSTM are calculated per frame, and the last k activations are used to create a fixed feature vector $\mathbf{f}$. Note that since we have three enrollment templates, we can keep all the three as separate templates, or average them into one single vector. We will show in our experiments of how template averaging impacts performance.

At runtime, another LSTM feature vector is generated in the same way from a sliding window, and Cosine distance is used to measure the similarity between the keyword template(s) and the sliding window. Decisions are made based on the similarity score.

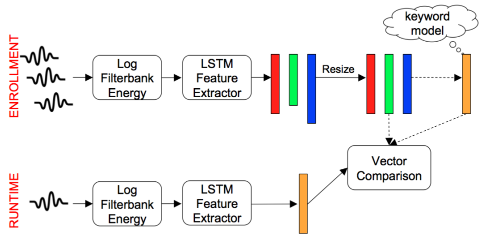

Fig. 2. Framework of the LSTM KWS system.

3. DTW BASELINE SYSTEM

DTW QbyE systems usually consist of two steps. First, features are extracted at the frame level [17, 18, 19]. Second, DTW is performed to compare the feature matrix of the templates and the test segment. A confidence score is then computed from the DTW alignment cost.

We use phoneme posteriorgram features in our implementation. Specifically, we label the training data by taking the forced-aligned 14,336 context-dependent (CD) state labels, and remapping this to a set of 43 phonemes, including the silence phone. We then train a neural network (i.e., DNN, LSTM) with 43 phoneme outputs. After training, each input frame of the template and test segments is converted to a posteriorgram by passing this frame through the network. Another way of generating phoneme posteriorgrams is to train a network to predict all 14,336 CD states, and then map them to

43 phonemes when computing the posteriorgrams. However, this greatly increases the number of model parameters due to the large output layer.

A full picture of the baseline DTW KWS system is shown in Figure 3. In the enrollment phase, log filterbank energy features are extracted at the frame level for the three keyword examples, which are then fed to the neural network to generate keyword phoneme posteriorgrams. In the runtime phase, sliding window phoneme posteriorgrams are computed in the same way, and DTW is performed to compare the sliding window posteriorgrams and the keyword posteriorgrams. Detection decisions are then made based on the DTW alignment scores.

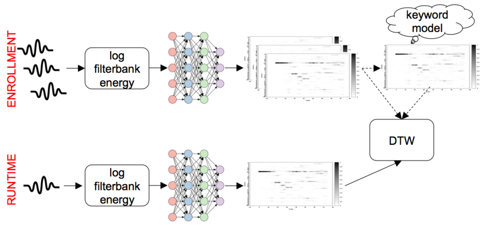

Fig. 3. Framework of the baseline DTW KWS system.

4. TEMPLATE AVERAGING

It has been shown in [18] that increasing the number of enrollment templates usually leads to performance improvements. In our work, we use three enrollment templates.

There are various techniques to perform evaluation with a test utterance given multiple enrollment templates. For example, in [18], the test audio is compared against each of the enrollment templates, generating one score for each template. These scores are then merged into one final score for decision making. Evaluating against three templates also increases the computational cost, which is not favored by on-device applications. Therefore, we propose to combine multiple templates into a single template for scoring.

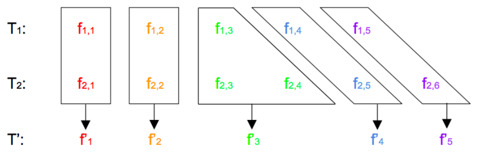

Fig. 4. Example of template averaging.

Figure 4 shows how we combine templates. We use two templates in the figure for simplicity, where T_1 is the first template, and T_2 is the second template. We first align the second template T_2 to the first template T_1. In the DTW KWS system, since the templates T_1 and T_2 have different length, we use DTW to align them. Figure 4 shows the aligned frames in a box. In the LSTM KWS system, however, templates are fixed length, so we align them by their dimensionality index. Once frames are aligned, we compute the combined frame by averaging them. For example, $f'_3 = (f_{1,3} + f_{2,3} + f_{2,4})/3$. The combined template T' will be used as the only template at runtime. If more templates are available, we always combine the new template with the previous combined one.

The proposed template averaging method relies heavily on the quality of the first template. In our experiments we ignore this effect and simply choose the first template randomly. Another option is to choose the longest template as the first one, as proposed in [25].

5. EXPERIMENTAL SETUP

5.1. Keywords for Evaluation

Table 1 lists keywords used in our experiments. Our test-set combines anonymized voice search queries as negative examples, and utterances including these keywords as positive examples.

Table 1. Keywords used in evaluation

hello genie	hi galaxy
okay glass	change watch face
open settings	show agenda
show alarms	show step count

We build one keyword model for each (speaker, keyword) combination. It is out of scope for this paper to evaluate impostor performance (same keyword, different speaker), so positive examples only include utterances from that selected speaker and keyword.

Results are reported in the form of a modified version of receiver operating characteristic (ROC) curves [1], lower curves are better. False alarm and false rejection counts are collected from all the models to compute the false rejection rate at a certain false alarm rate. The curve is obtained by sweeping the decision threshold.

5.2. Training

The neural network models are trained with 2,500 hours of speech data, anonymized and manually transcribed. This dataset does not have any assumption on the keywords that will be evaluated. All models are trained using the cross-entropy criterion, with asynchronous stochastic gradient descent (ASGD) [26].

The enrollment set contains 3 keyword examples for each (speaker, keyword) combination. The evaluation set has 9k positive examples for the 8 selected keywords, and 36k negative examples. We use many more negative than positive examples to simulate the expected application usage.

6. RESULTS

6.1. Initial Results

Figure 5 shows the performance of the proposed LSTM Feature Extractor and 2 DTW KWS systems. The LSTM Feature Extractor system takes a 40-dimensional input feature and has 2 LSTM layers, each with 128 cells, resulting in 152k parameters. For the DTW KWS systems, we compare obtaining posteriors from a DNN and an LSTM. The LSTM-posterior model also takes a 40-dimensional input feature, followed by 2 LSTM layers, each with 128 cells, and 43 phoneme output targets, resulting in 157k total parameters. The DNN-posterior model has 5 layers, each with 128 hidden units, and 43 phoneme targets. It uses a 40-dimensional feature with 10 history frames and 5 future frames, resulting in a DNN with 152k parameters. Those network topologies are mainly chosen to match each other's parameter size, and are not explicitly tuned for performance.

We first use clean enrollment and evaluation data. In Figure 5, at roughly the same parameter size, the *Phone LSTM + DTW* system outperforms the *Phone DNN + DTW* system. The proposed *LSTM Feat Extractor* system improves significantly over both DTW systems. At 0.5% false alarm rate, which is the desired operating point in our application, the *LSTM Feat Extractor* system yields 86% and 88% relative improvements respectively over the DTW systems.

We also explore the robustness of the proposed KWS approach by adding 10 dB of babble noise to the evaluation set. Results in

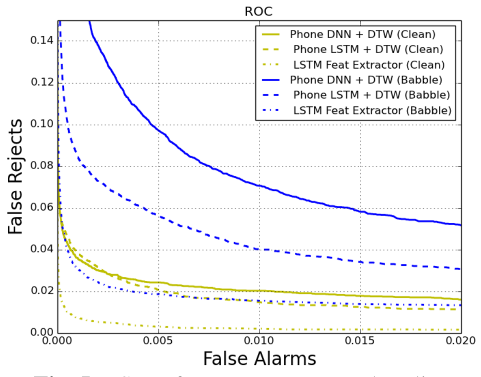

Fig. 5. LSTM feature extractor vs. baseline

Figure 5 show a performance degradation for all the systems due to noise. However, the general story does not change: the *Phone LSTM + DTW (Babble)* system is better than the *Phone DNN + DTW (Babble)* system (a lot better in this noisy case), while the *LSTM Feat Extractor (Babble)* system outperforms both of them.

6.2. Template Averaging

Figure 6 compares the performance with and without template averaging. In this figure, the dashed curves are generated with template averaging, i.e., all the three templates are averaged into one template, and is used as the only template during runtime. The solid curves are generated without template averaging. Figure 6 suggests that template averaging does not degrade performance significantly. This is good news for on-device applications since we can reduce the runtime computation greatly (3x) without hurting the performance.

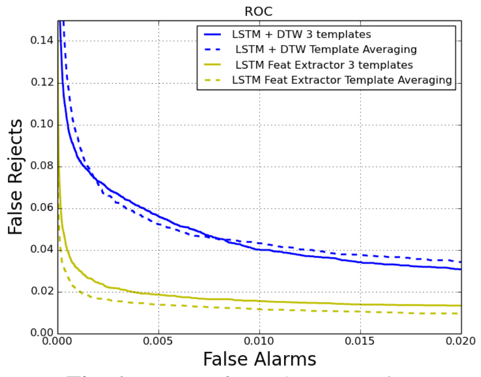

Fig. 6. Impact of template averaging

6.3. Whole Word Modeling

In Figure 7 we evaluate the importance of training the LSTM feature extractor with whole word output targets. The *Phone LSTM + DTW (Babble)* and *Word LSTM Feat Extractor (Babble)* curves are the same curves from Figure 5. For the *Phone LSTM Feat Extractor (Babble)* curve, we take the Phone LSTM model, remove the output layer and treat it as a feature extractor for keyword spotting. The performance of this system is much worse than the original LSTM feature extractor trained with whole word output targets, and it is also worse than the *Phone LSTM + DTW* system where we take the LSTM from. This implies that whole word modeling is critical in our LSTM KWS system.

6.4. Noisy Enrollment

Finally, we show the robustness of our proposed KWS approach when adding different noise sources at enrollment and evaluation

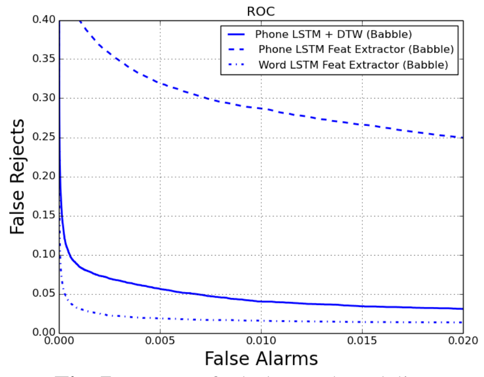

Fig. 7. Impact of whole word modeling

time. To simulate this, we add 10 dB of babble noise to the clean evaluation data, and also add a different 10 dB cafe noise to the clean enrollment data, so that the keyword models built from the noisy enrollment data are corrupted. As shown in Figure 8, the performance degrades dramatically for all the systems. For example, at 0.5% (0.005) false alarm rate, the two DTW KWS systems both give 99.8% false rejection, basically rejecting everything, while the LSTM KWS system gives a false rejection rate of 63%.

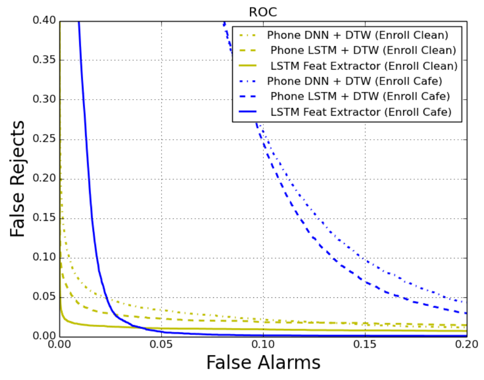

Fig. 8. Impact of noisy enrollment

7. CONCLUSION AND FUTURE WORK

We presented a LSTM feature extractor for the QbyE task in KWS. Experimental results showed that our method outperformed the standard phoneme posteriorgram + DTW system. We also proposed a template averaging technique, which allows us to combine multiple templates into one without performance degradation on our dataset. This technique is especially important for on-device applications since it can reduce the runtime computation cost. We tested the proposed QbyE system along with the baseline systems on various enrollment/evaluation conditions, and showed that it is important to have a clean enrollment environment for QbyE tasks.

As a future direction, we will look into ways to further improve our LSTM feature extractor. For example, dimensionality reduction can be applied to reduce the stacked feature vector. We will also compare our method with the latest DTW QbyE systems, as described in [25, 27].

8. ACKNOWLEDGEMENTS

We gratefully acknowledge many fruitful discussions with all the members from Google's Mobile Speech team. We especially would like to thank Rohit Prabhavalkar for his valuable suggestions and Oriol Vinyals for his help with LSTM.

9. REFERENCES

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

[2] J. Schalkwyk, D. Beeferman, F. Beaufays, B. Byrne, C. Chelba, M. Cohen, M. Kamvar, and B. Strope, "Your word is my command: Google search by voice: a case study," in *Advances in Speech Recognition*. Springer, 2010, pp. 61–90.

[3] J. R. Rohlicek, W. Russell, S. Roukos, and H. Gish, "Continuous hidden Markov modeling for speaker-independent word spotting," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 1989, pp. 627–630.

[4] R. C. Rose and D. B. Paul, "A hidden Markov model based keyword recognition system," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 1990, pp. 129–132.

[5] J. Wilpon, L. Miller, and P. Modi, "Improvements and applications for key word recognition using hidden Markov modeling techniques," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 1991, pp. 309–312.

[6] M.-C. Silaghi and H. Bourlard, "Iterative posterior-based keyword spotting without filler models," in *Proceedings of the Automatic Speech Recognition & Understanding (ASRU) Workshop*. Citeseer, 1999, pp. 213–216.

[7] M.-C. Silaghi, "Spotting subsequences matching an HMM using the average observation probability criteria with application to keyword spotting." in *Proceedings of the National Conference on Artificial Intelligence*, vol. 20. AAAI Press/MIT Press, 2005, p. 1118.

[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 *Proceedings of INTERSPEECH*. ISCA, 2007, pp. 314–317.

[9] S. Parlak and M. Saraclar, "Spoken term detection for Turkish broadcast news," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2008, pp. 5244–5247.

[10] G. Chen, S. Khudanpur, D. Povey, J. Trmal, D. Yarowsky, and O. Yilmaz, "Quantifying the value of pronunciation lexicons for keyword search in low resource languages," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2013.

[11] J. Mamou, B. Ramabhadran, and O. Siohan, "Vocabulary independent spoken term detection," in *Proceedings of SIGIR*. ACM, 2007, pp. 615–622.

[12] R. G. Wallace, R. J. Vogt, and S. Sridharan, "A phonetic search approach to the 2006 NIST spoken term detection evaluation," in *Proceedings of INTERSPEECH*. ISCA, 2007.

[13] M. Akbacak, D. Vergyri, and A. Stolcke, "Open-vocabulary spoken term detection using graphone-based hybrid recognition systems," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2008, pp. 5240–5243.

[14] G. Chen, O. Yilmaz, J. Trmal, D. Povey, and S. Khudanpur, "Using proxies for OOV keywords in the keyword search task," in *Proceedings of the Automatic Speech Recognition and Understanding (ASRU) Workshop*. IEEE, 2013, pp. 416–421.

[15] C. Parada, A. Sethy, and B. Ramabhadran, "Query-by-example spoken term detection for OOV terms," in *Proceedings of the Automatic Speech Recognition & Understanding (ASRU) Workshop*. IEEE, 2009, pp. 404–409.

[16] H. Soltau, G. Saon, and B. Kingsbury, "The IBM Attila speech recognition toolkit," in *Proceedings of Spoken Language Technology Workshop (SLT)*. IEEE, 2010, pp. 97–102.

[17] K. Levin, K. Henry, A. Jansen, and K. Livescu, "Fixed-dimensional acoustic embeddings of variable-length segments in low-resource settings," in *Proceedings of the Automatic Speech Recognition & Understanding (ASRU) Workshop*. IEEE, 2013, pp. 410–415.

[18] Y. Zhang and J. R. Glass, "Unsupervised spoken keyword spotting via segmental DTW on Gaussian posteriorgrams," in *Proceedings of the Automatic Speech Recognition & Understanding (ASRU) Workshop*. IEEE, 2009, pp. 398–403.

[19] T. J. Hazen, W. Shen, and C. White, "Query-by-example spoken term detection using phonetic posteriorgram templates," in *Proceedings of the Automatic Speech Recognition & Understanding (ASRU) Workshop*. IEEE, 2009, pp. 421–426.

[20] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[21] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2013, pp. 6645–6649.

[22] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proceedings of Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2014, pp. 1701–1708.

[23] T. Hughes and K. Mierle, "Recurrent neural networks for voice activity detection," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2013, pp. 7378–7382.

[24] X. Lei, A. Senior, A. Gruenstein, and J. Sorensen, "Accurate and compact large vocabulary speech recognition on mobile devices," in *Proceedings of INTERSPEECH*. ISCA, 2013.

[25] L. J. Rodriguez-Fuentes, A. Varona, M. Penagarikano, G. Bordel, and M. Diez, "High-performance query-by-example spoken term detection on the SWS 2013 evaluation," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 7819–7823.

[26] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le *et al.*, "Large scale distributed deep networks," in *Proceedings of Advances in Neural Information Processing Systems*, 2012, pp. 1223–1231.

[27] I. Szöke, M. Skácel, and L. Burget, "BUT QUESST 2014 system description," in *Proceedings of CEUR Workshop*, vol. 2014, no. 1263. CEUR-WS.org, 2014, pp. 1–2.