THE THUEE SYSTEM FOR THE OPENKWS14 KEYWORD SEARCH EVALUATION

Meng Cai, Zhiqiang Lv, Beili Song, Yongzhe Shi, Weilan Wu, Cheng Lu, Wei-Qiang Zhang and Jia Liu

Tsinghua National Laboratory for Information Science and Technology
Department of Electronic Engineering, Tsinghua University, Beijing 100084, China

{cai-m10,lv-zq12,songbl12}@mails.tsinghua.edu.cn,
shiyz09@gmail.com, wuweilan07@foxmail.com, lucheng1983@163.com,
{wqzhang,liuj}@tsinghua.edu.cn

ABSTRACT

The OpenKWS14 keyword search evaluation is one of the most challenging and influential evaluations in the field of speech recognition. Its goal is to build a high-performance keyword search system for a minority language with limited training data in a short period of time. We present the system of the Department of Electronic Engineering, Tsinghua University (THUEE team) for the OpenKWS14 keyword search evaluation. The highlights of the system include the use of convolutional maxout neural networks for acoustic modeling and the use of neural network language models for one-pass lattice generation. The final system is a fusion of 8 subsystems. The system has achieved an actual term weighted value (ATWV) of 0.5107 for the full language pack (FullLP) condition in the evaluation, ranking third among the participating teams.

Index Terms— Acoustic Modeling, Language Modeling, Deep Neural Network, Keyword Spotting, Low-Resource

1. INTRODUCTION

Automatic speech recognition (ASR) technology is developing fast these days, largely due to the widespread application of deep neural networks for acoustic modeling since 2011 [1, 2, 3]. While the speech recognition error rates for majority languages are relatively low, the performances for minority languages remain poor since there are much less training data compared with majority languages. With so many minority languages in the world, building a speech recognition system with limited data resources is a major challenge for ASR.

The OpenKWS evaluation is part of the IARPA Babel program. Its goal is to build a high-performance keyword search system for a "surprise" language with limited training data in a short period of time. The identity of the language for the OpenKWS14 evaluation is Tamil. According to the evaluation plan [4], there are several evaluation conditions with respect to the build pack utilization, the language resources utilization and whether to re-use the test audio. We focus on the required condition for all the participants, under which we use the full language pack (FullLP), the baseline language resources (BaseLR) and there is no test audio re-use (NTAR). The evaluation metric for the keyword search accuracy is actual term-weighted value (ATWV), which is defined as

$$\mathrm{ATWV} = 1 - \frac{1}{K}\sum_{w=1}^{K}\left(\frac{\#miss(w)}{\#ref(w)} + \beta\frac{\#fa(w)}{T - \#ref(w)}\right), \quad (1)$$

where K is the number of keywords occurred in the reference, $\#ref(w)$ is the number of reference occurrences of the keyword w, $\#miss(w)$ is the number of missing keywords, $\#fa(w)$ is the number of false alarms, β is a constant set at 999.9 and T is the total duration of the corpus in seconds, as a proxy of the total number of trials. The higher the ATWV the better the performance.

Previous NIST spoken term detection evaluations show that the ASR framework works well for keyword spotting [5]. The ASR framework is also adopted by all the participants in the previous OpenKWS13 evaluation and in the Babel program [6, 7, 8, 9]. We present the detailed system description of the THUEE team, hoping to provide some inspirations to not only the OpenKWS participants, but also researchers who work on ASR and keyword search tasks.

The remainder of this paper is organized as follows. In Section 2, we present the acoustic modeling strategies of the system. In Section 3, we show our language modeling strategies. In Section 4, we introduce our decoding and keyword searching strategies. The detailed experiments are presented in Section 5. The conclusions and future works are given in Section 6.

2. ACOUSTIC MODELING

We have two acoustic modeling systems. One is based on the Kaldi toolkit [10]. The other is based on HTK [11] but we implement the DNN acoustic modeling tools in-house. Our

This work is supported by National Natural Science Foundation of China under Grant No. 61273268, No. 61370034, No. 61403224 and No. 61005017.

in-house DNN training tool support fully-connected sigmoid DNNs, deep maxout neural networks [12], stochastic pooling maxout neural networks [13], convolutional neural networks (CNN) [14] and convolutional maxout neural networks (CMNN) [15]. We further describe some details of our in-house acoustic models, namely the maxout neural networks, the CNNs and the CMNNs.

The maxout neural network is a feed-forward neural network. The major difference from conventional sigmoid neural network is that the neuron nonlinearity is achieved by selecting the maximum value within a local region, i.e.,

$$h_l^i = \max_{j \in 1...k} z_l^{ij}, \tag{2}$$

where h_l^i is the ith activation in the lth layer, $\{z_l^{ij} | j \in 1...k\}$ is the piece group corresponding to the activation h_l^i. Previous works [12, 16] have demonstrated that maxout neurons are effective for acoustic modeling under low-resource conditions. One explanation is that the maxout nonlinearity avoids the vanishing gradient problem [17] and yields better optimization performance.

The CNN is a powerful model that shows excellent performance for acoustic modeling [18, 14]. There are three major differences between CNN and conventional sigmoid neural networks. First, the CNN applies a set of small weights to local regions of speech features, so that the local properties of speech (e.g. the formant) are explicitly modeled. Second, the value of the local weights are often shared in CNN, thus parameter estimation can be more robust than conventional DNNs. Third, there is often a pooling layer after the convolutional layer in CNN, making the output of the pooling layer less sensitive to the variance of the input features.

As the maxout neural network and the CNN are both effective models for low-resource speech recognition, it is natural to consider the possibility of combining them. The convolutional maxout neural network (CMNN) takes advantage of both models. There are three building blocks of the CMNN, namely, the convolutional structure, the maxout nonlinearity and the dropout training strategy [19]. In the CMNN, all the neurons are maxout neurons. The lower layers of the CMNN are convolutional while the upper layers are fully-connected. It is also pointed out in [15] that the application of dropout to only the upper fully-connected layers of the CMNN is effective for acoustic modeling in low-resource conditions.

We will show in the experiments the performance of the acoustic models for the OpenKWS14 task.

3. LANGUAGE MODELING

Language modeling is also an important issue in ASR. In addition to a baseline trigram language model, we also use a feed-forward neural network language model (NNLM) [20]. Because NNLMs are computationally expensive, previously NNLMs have only been used to rescore lattices or n-best lists. We have proposed a method to incorporate the NNLM directly into the decoder for first-pass lattice generation [21].

In NNLM, the output layer is a softmax layer to generate the probabilities, i.e.,

$$P(q(w_t) = j | w_{t-(N-1)}^{t-1}) = \frac{\exp(o_{tj})}{z_t}, \tag{3}$$

where w_t stands for the tth word, $w_{t-(N-1)}^{t-1}$ stands for the $N-1$ words before w_t, $q(w_t)$ is the word index. o_{tj} is the value before the softmax function and $z_t = \sum_{k=1}^{|\mathbf{V}|} \exp(o_{tk})$ is the normalizing factor for the softmax function. As the number of output neurons for the NNLM equals the vocabulary size $|\mathbf{V}|$, which is quite large, the most time consuming computation is to obtain the normalizing factor z_t.

Our solution to speedup the NNLM computation is to regularize the variance of the normalizing factor z_t. We intend to minimize the variance of z_t. If the variance of z_t tends to become zero, the normalizing factor z_t will tend to become a constant. Then only o_{tk} needs to be computed while z_t can be omitted. The way to achieve this is to add a variance regularization term to the cross entropy function $J(\Theta)$. Thus the optimization function becomes

$$\widetilde{J}(\Theta) = J(\Theta) + \frac{\eta}{2} \frac{1}{|\mathbf{T}|} \sum_{t=1}^{|\mathbf{T}|} (\log(z_t) - \frac{1}{|\mathbf{T}|} \sum_{t=1}^{|\mathbf{T}|} \log(z_t))^2, \tag{4}$$

where η is a parameter to control the regularization term. The gradient of $\widetilde{J}(\Theta)$ is not difficult to derive. Then the NNLM is trained by conventional back-propagation with the gradient of the regularization term.

Since the NNLM trained with the regularization term is much more computationally efficient at the test phase, we can incorporate it directly into the decoder for first-pass lattice generation. The implementation details are presented in [21].

4. DECODING AND KEYWORD SEARCHING

In the OpenKWS14 evaluation, we use the Kaldi decoder for the Kaldi-based acoustic model and the trigram language model, and we use our in-house decoder for our in-house acoustic models and NNLM.

Our in-house decoder is based on HDecode in the HTK tools. It is a highly-optimized dynamic decoder [22]. The lattices generated by the decoder are then converted to confusion networks [23] for keyword searching.

We handle the out-of-vocabulary (OOV) keywords using a similar idea to [24]. The pronunciations of the OOVs are estimated using the grapheme-to-phoneme (G2P) method [25]. For each OOV keyword, a proxy in-vocabulary (IV) keyword is selected based on the word frequency in the training corpus.

We also discover in our experiments that the score normalization and system combination are crucial for the final

ATWV. We use the keyword-specific thresholding and exponential normalization (KST) [26] to normalize the scores of all the sub-systems. The system combination is performed using the CombMNZ method [27]. The weights of the subsystem scores for the system combination are based on their maximum term-weighted values (MTWVs).

5. EXPERIMENTS

5.1. The OpenKWS14 corpus

The OpenKWS14 corpus[1] includes a training set, a development set and an evaluation set. The training set in the fullLP language pack contains about 63 hours of transcribed conversational audio (excluding silence) and about 12 hours of untranscribed conversational audio. Most of the conversational audio is A-law telephone speech with a sample rate of 8kHz but some conversational audio is recorded at a sample rate of 48kHz. There is also about 16 hours of scripted audio in the training set which contains read-style speech. The development set contains about 10 hours of conversational speech and the evaluation set contains about 93 hours of conversational speech. The training and the development set are released a month before sites submit system outputs to NIST, while the evaluation set is released a week before result submission.

A pronunciation lexicon is released along with the training set. There are 34 non-silence phonemes and the vocabulary size for the conversational training set in the fullLP language pack is 58470.

5.2. System setup

The baseline of our in-house system is based on HTK. We only use the 63 hours of transcribed conversational training data for the baseline setup. We extract 13-dimensional PLP features along with their first-, second- and third-order derivatives. A GMM-HMM acoustic model is trained with the 52-dimensional PLP features. The dimension of the features is then reduced to 39 using HLDA. The GMM-HMM contains 4505 triphone states via decision tree clustering and 20 Gaussian mixtures for each state. The GMM-HMM is trained with the maximum-likelihood criterion.

The baseline language model is a trigram model with modified KN smoothing. The transcription of the conversational training data is used as the training text for the trigram. The perplexity of the language model on the development text is 872.571.

The baseline GMM-HMM is used to generate the state-level transcriptions of the training data via forced alignment. Then we start DNN training with the cross entropy criterion. The features for the DNN training are 40-dimensional filter-bank features with first- and second-order derivatives. A context window of 11 frames (5+1+5) is used so that the dimension of the input layer for DNN is 1320. We use 6 hidden layers and 1500 nodes for each hidden layer. The DNN output layer size is the same as the GMM-HMM, which is 4505. The training of DNN starts with layer-wise RBM pre-training [2] and then with standard back-propagation. After training converges, the DNN is used to re-generate the state-level transcriptions and we train another DNN with the new transcriptions. The results of the baseline models on the development set are given in Table 1.

[1] language collection release `IARPA-babel204b-v1.1b`.

Table 1. Word error rate (WER) (%) of baseline systems on the development set.

System	WER
GMM-HMM	81.5
DNN-HMM	70.1
DNN-HMM + re-alignment	69.0

Table 2. Word error rate (WER) (%) and MTWV of the DNN, CNN, CMNN and the bottleneck (BN) model on the development set. The MTWVs are evaluated with the initial BBN keywords and the scores are normalized using KST.

System	WER	MTWV
S1: DNN	69.0	0.2071
S2: CNN	68.1	0.2215
S3: CMNN	67.1	0.2316
S4: CMNN BN + maxout	66.6	0.2238

5.3. Results of DNN, CNN and CMNN

As it is shown in Table 1, the baseline results are poor for Tamil. We then try the CNN and the CMNN acoustic models as they both have shown superior performance over conventional DNNs for other tasks [14, 15]. The features for the CNN and the CMNN are the same as that for the DNN, namely 120-dimensional filter-bank features.

The CNN contains two convolutional hidden layers and a max-pooling layer. The input feature vector is divided into 40 bands. The corresponding dimension of the 11 consecutive feature frames are arranged in each band, together with their derivatives. So that the input dimension to the CNN is 33×40. The first set of convolutional filters are applied to 8 consecutive bands and generate 768 feature mappings. So the number of bands for the second hidden layer is $40 - 8 + 1 = 33$. We then apply max-pooling across 3 bands to generate 11 bands. The second set of convolutional filters are applied to 4 consecutive bands and generate 512 feature mappings. Three fully-connected hidden layers of 1500 nodes are arranged after the convolutional layers. The total number of parameters for the CNN is 18.5M.

The input features for the CMNN are arranged in the same way as the CNN. The first set of convolutional filters are ap-

Table 3. MTWVs of the Kaldi-based models on the development set. The MTWVs are evaluated with the initial BBN keywords and the scores are normalized using KST.

System	MTWV
S5: SGMM	0.2176
S6: BN + DNN	0.2269
S7: BN + pitch + DNN	0.2373
S8: fbank + pitch + DNN + sMBR	0.2487

Table 4. MTWVs with and without the NNLM on the development set. The MTWVs are evaluated with the initial BBN keywords and the scores are normalized using KST.

System	MTWV
S3: CMNN	0.2316
S9: CMNN + NNLM	0.2386

plied to 8 consecutive input bands and generate 512 feature mappings. We then apply the maxout nonlinearity of 2 pieces and max-pooling of 3 bands. The second set of convolutional filters are applied to 4 bands and generate 512 features mappings. There are 6 fully connected hidden layers following the convolutional layers, all of them have 1000 maxout neurons with 2 pieces. We apply a dropout rate of 0.2 to the 6 fully-connected hidden layers of the CMNN. The total number of parameters for the CMNN is 16.5M.

We also try to use the CMNN to extract the bottleneck (BN) features. The structure of the CMNN is similar to the previous one except we add a 150-dimensional linear BN layer before the output layer. The number of nodes for the output layer is 512 via tree clustering. The BN features are then fed to a maxout neural network, which have 9 hidden layers, 1000 neurons for each layer and 2 pieces for each neuron.

The CNN, the CMNN and the BN model all use the state-level transcriptions generated by the baseline DNN. Their development set results are given in Table 2.

5.4. Results of Kaldi-based system

We have built 4 systems based on the Kaldi toolkit. Their MTWVs are shown in Table 3. The SGMM system [28] is trained with PLP features plus pitch features [29]. It is also trained with speaker adaptive training (SAT) and bMMI discriminative training. There are two DNN models trained with bottleneck features. One is built upon the PLP plus pitch features while the other is trained only with PLP features. The best-performing model is a DNN with filter-bank plus pitch features. It is first trained with the cross entropy criterion and then enhanced by sMBR sequence training [30]. From the results in Table 3 we see that the pitch features are very effective for Tamil although it is not a tonal language. Also, sequence training provides large gains for the keyword search task.

Table 5. System combination results on the development set.

System	MTWV	
	w/ KST	w/o KST
S1-S9	0.3081	0.2915
S1-S8	0.3082	0.2962
S2-S9	0.3072	0.2929
S1 + S2 + S4-S9	0.3120	0.2935

Table 6. Comparing the evaluation data set ATWVs under the fullLP condition with other OpenKWS14 participants.

Participants	ATWV
LORELEI	0.5802
SINGA	0.5326
THUEE	0.5107
MSIIP	0.5104

5.5. Results of NNLMs

We then experiment with the neural network language model. The NNLM is trained with the transcriptions of the 63-hour conversational training data. The context length of the NNLM is 4 and the dimensions of the word feature and the hidden layer are both set to 300. The NNLM is trained with our proposed variance regularization, where the control parameter η is empirically set to 2.5. We try the NNLM with the CMNN acoustic model and our in-house decoder. The results are shown in Table 4. The results show that the NNLM is superior than the trigram for this keyword search task.

5.6. Results of system combination

It is well-known that system combination can greatly improve the final keyword spotting result [26, 27]. Our system combination results are shown in Table 5. The best result is obtained by combining 8 sub-systems. The results also show that applying KST normalization after system combination is helpful, although KST is already applied to every sub-system. Table 6 shows the official results on the evaluation set released by NIST. The ATWV of 0.5107 is a state-of-the-art result.

6. CONCLUSIONS AND FUTURE WORKS

We propose a state-of-the-art keyword search system that achieved a good performance in the OpenKWS14 evaluation. Our main discoveries include: (1) CNN and maxout neurons are effective for low-resource acoustic modeling. (2) NNLMs can be useful for keyword spotting. (3) Pitch features and DNN sequence training produce great gains for keyword spotting. (4) System combination is crucial for the final result. In the future, we plan to explore efficient DNN sequence training algorithms and better ways to handle OOVs.

7. REFERENCES

[1] F. Seide, G. Li, and D. Yu, "Conversational speech transcription using context-dependent deep neural networks," in *Proc. Interspeech*. Florence, Italy, 2011, pp. 437–440.

[2] G.E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large vocabulary speech recognition," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 20, pp. 30–42, January 2012.

[3] G.E. Hinton, L. Deng, D. Yu, G.E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T.N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, pp. 82–97, November 2012.

[4] NIST, "Draft KWS14 keyword search evaluation plan," http://nist.gov/itl/iad/mig/upload/KWS14-evalplan-v11.pdf, 2014.

[5] J.G. Fiscus, J. Ajot, J.S. Garofolo, and G. Doddingtion, "Results of the 2006 spoken term detection evaluation," in *Proc. ACM SIGIR Workshop on Searching Spontaneous Conversational*, 2007, pp. 51–57.

[6] S. Tsakalidis, R. Hsiao, D. Karakos, T. Ng, S. Ranjan, G. Saikumar, L. Zhang, L. Nguyen, R. Schwartz, and J. Makhoul, "The 2013 BBN Vietnamese telephone speech keyword spotting system," in *Proc. ICASSP*. Florence, Italy, 2014, pp. 7879–7883.

[7] X. Cui, B. Kingsbury, J. Cui, B. Ramabhadran, A. Rosenberg, M.S. Rasooli, O. Rambow, N. Habash, and V. Goel, "Improving deep neural network acoustic modeling for audio corpus indexing under the IARPA Babel program," in *Proc. Interspeech*. Singapore, 2014, pp. 2103–2107.

[8] M. Karafiat, F. Grezl, K. Vesely, M. Hannemann, I. Szoke, and J.H. Cernocky, "BUT 2014 Babel system: Analysis of adaptation in NN based systems," in *Proc. Interspeech*. Singapore, 2014, pp. 3002–3006.

[9] N.F. Chen, S. Sivadas, B.P. Lim, H.G. Ngo, H. Xu, V.T. Pham, B. Ma, and H. Li, "Strategies for Vietnamese keyword search," in *Proc. ICASSP*. Florence, Italy, 2014, pp. 4149–4153.

[10] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The Kaldi speech recognition toolkit," in *Proc. ASRU*, 2011.

[11] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, X. Liu, G. Moore, J. Odell, D. Ollason, Povey D., Valtchev V., and Woodland P., *The HTK book, version 3.4.1*, Cambridge University Engineering Department, 2009.

[12] M. Cai, Y. Shi, and J. Liu, "Deep maxout neural networks for speech recognition," in *Proc. ASRU*. Olomouc, Czech Republic, 2013, pp. 291–296.

[13] M. Cai, Y. Shi, and J. Liu, "Stochastic pooling maxout networks for low-resource speech recognition," in *Proc. ICASSP*. Florence, Italy, 2014, pp. 3290–3294.

[14] T. Sainath, A. Mohamed, B. Kingsbury, and B. Ramabhadran, "Deep convolutional neural networks for LVCSR," in *Proc. ICASSP*. Vancouver, Canada, 2013, pp. 8614–8618.

[15] M. Cai, Y. Shi, J. Kang, J. Liu, and T. Su, "Convolutional maxout neural networks for low-resource speech recognition," in *Proc. ISCSLP*. Singapore, 2014, pp. 133–137.

[16] Y. Miao, F. Metze, and S. Rawat, "Deep maxout networks for low-resource speech recogntion," in *Proc. ASRU*. Olomouc, Czech Republic, 2013, pp. 398–403.

[17] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. AISTATS*, 2010, pp. 249–256.

[18] O. Abdel-Hamid, A. Mohamed, H. Jiang, and G. Penn, "Applying convolutional neural networks concept to hybrid NN-HMM model for speech recogntion," in *Proc. ICASSP*. Kyoto, Japan, 2012, pp. 4277–4280.

[19] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R.R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv:1207.0580*, 2012.

[20] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journel of Machine Learning Research*, vol. 3, pp. 1137–1155, 2003.

[21] Y. Shi, W.Q. Zhang, M. Cai, and J. Liu, "Efficient one-pass decoding with NNLM for speech recognition," *IEEE Signal Processing Letters*, vol. 21, no. 4, pp. 377–381, Apr. 2014.

[22] Y. Shan, X. Chen, Y. Shi, and J. Liu, "Fast language model look-ahead algorithm using extended n-gram model," *Acta Automatica Sinica*, vol. 38, no. 10, pp. 1618–1626, 2012.

[23] L. Mangu, E. Brill, and A. Stolcke, "Finding consensus in speech recognition: Word error minimization and other applications of confusion networks," *Computer Speech and Language*, vol. 14, pp. 373–400, 2000.

[24] G. Chen, O. Yilmaz, J. Trmal, D. Povey, and S. Khudanpur, "Using proxies for OOV keywords in the keyword search task," in *Proc. ASRU*. Olomouc, Czech Republic, 2013, pp. 416–421.

[25] M. Bisani and H. Ney, "Joint-sequence models for grapheme-to-phoneme conversion," *Speech Communication*, vol. 50, pp. 434–451, 2008.

[26] D. Karakos, R. Schwartz, S. Tsakalidis, L. Zhang, S. Ranjan, T. Ng, R. Hsiao, G. Saikumar, I. Bulyko, L. Nguyen, J. Makhoul, F. Grezl, M. Hannemann, M. Karafiat, I. Szoke, K. Vesely, L. Lamel, and V. Le, "Score normalization and system combination for improved keyword spotting," in *Proc. ASRU*. Olomouc, Czech Republic, 2013, pp. 210–215.

[27] J. Mamou, J. Cui, X. Cui, M. Gales, B. Kingsbury, K. Knill, L. Mangu, D. Nolden, M. Picheny, B. Ramabhadran, R. Schluter, A. Sethy, and P.C. Woodland, "System combination and score normalization for spoken term detection," in *Proc. ICASSP*. Vancouver, Canada, 2013, pp. 8272–8276.

[28] D. Povey, L. Burget, M. Agarwal, P. Akyazi, F. Kai, A. Ghoshal, O. Glembek, N. Goel, M. Karafiat, A. Rastrow, R.C. Rose, P. Schwarz, and S. Thomas, "The subspace Gaussian mixture model - a structured model for speech recognition," *Computer Speech and Language*, vol. 25, pp. 404–439, 2011.

[29] P. Ghahremani, B. BabaAli, and D. Povey, "A pitch extraction algorithm tuned for automatic speech recognition," in *Proc. ICASSP*. Florence, Italy, 2014, pp. 2513–2517.

[30] K. Vesely, A. Ghoshal, L. Burget, and D. Povey, "Sequence-discriminative training of deep neural networks," in *Proc. Interspeech*. Lyon, France, 2013, pp. 2345–2349.