Unsupervised Adaptation for Deep Neural Network using Linear Least Square Method

Roger Hsiao, Tim Ng, Stavros Tsakalidis, Long Nguyen and Richard Schwartz

Raytheon BBN Technologies
10 Moulton Street, Cambridge, MA 02138, USA
whsiao@bbn.com

Abstract

In this paper, we propose a novel model based adaptation for deep neural networks based on a linear least square method. Our proposed algorithm can perform unsupervised adaptation even if the auto transcripts may have 60-70% of word error rate. We evaluate our algorithm on low resource languages, from the the IARPA BABEL program, such as Assamese, Bengali, Haitian Creole, Lao and Zulu. Our experiments focus on unsupervised speaker, dialect and environment adaptation and we show that it can improve both speech recognition and keyword search performance.

Index Terms: deep neural network, unsupervised adaptation, keyword search

1. Introduction

Deep neural network (DNN) is widely used for acoustic modeling in automatic speech recognition [1, 2]. Technologies originally developed for Gaussian mixture model (GMM), like bottle-neck features, discriminative training have shown to be effective in improving DNN performance [3, 4, 5]. However, the application of model adaptation remains limited for DNN and it has been a popular research area recently [6, 7, 8, 9, 10].

In a broad term, adaptation aims to improve system performance by using a small amount of target data. The target data can be supervised, say enrollment data for a speech recognition system; or unsupervised, say an output from a preliminary system. The amount of data for adaptation is usually insufficient for full training. Instead, the data is used for adjusting the model parameters. For GMM, maximum likelihood linear regression (MLLR) [11] and maximum-a-posteriori (MAP) [12] are two widely used adaptation techniques, and they have been applied to speaker adaptation [11], environment adaptation [13], or language adaptation [14].

Adaptation for DNN is more challenging than GMM. One of the reasons is the sheer amount of parameters in a DNN. Estimating the DNN parameters with little data is a difficult task. Current popular approaches to DNN adaptation includes fine-tuning and I-vector based adaptation. Fine-tuning means adjusting the DNN parameters using some adaptation data [6, 7, 8]. In contrast, I-vector based adaptation seeks not to adjust the DNN parameters but using I-vectors as supplementary features to the DNN [9]. The work in [9, 10, 15] shows that speaker adaptation can be done by extracting I-vectors which contain rich speaker information.

There are advantages and disadvantages for these two approaches. For fine-tuning, [6, 7] have shown that it is effective for supervised adaptation. Unsupervised adaptation can be challenging for large DNN [7] and careful smoothing and regularization would help [8]. Nonetheless, current studies on fine-tuning for unsupervised adaptation mostly focus on the systems with relatively good initial error rate. If the error is high, fine-tuning may become more difficult. For adaptation using I-vectors, while it works on unsupervised data [9, 10], adaptation is restricted to the type that the training data supports. If one would like to perform channel adaptation with I-vectors, the training data should cover many different channels. Such diversity may not be available for the training data of some applications. Compared to the adaptation techniques available to GMM, DNN adaptation seems to have more restrictions.

In this work, we propose a versatile DNN adaptation algorithm which is comparable to the GMM adaptation techniques. Our goal is to design an algorithm which supports unsupervised adaptation even if the auto transcripts have high word error rate (WER) of 60-70%, At the same time, the type of adaptation is not limited by the diversity of the training data. Our approach is based on a linear least square method (LLS) originally proposed in [16] for single layer perceptron training with invertible activation functions. This approach is further explored in [17] with different regularization and extensions. In this paper, we extend the basic LLS algorithm for DNN and we also support commonly used activation functions. We evaluate the performance under the IARPA BABEL program, and we show the improvement to speech recognition and keyword search systems. In this paper, we also explore unsupervised environment, dialect and speaker adaptation using the proposed algorithm.

This paper is organized as follows: In section 2, we describe the original linear least square method and our extension to the algorithm. Section 3 contains experimental results and we conclude our work in section 4.

2. Linear Least Square Method for DNN Training

To facilitate discussion, we define the DNN using the following notions: a DNN may contain $N+2$ layers where the first layer (indexed with 0) is the input layer and the last layer (indexed with $N+1$) is the output layer. Layer 1 to N are the hidden layers and each layer contains K_i units. Each unit has an activation function. Mathematically, each layer of the DNN can be evaluated by the following equations,

$$z_t^i = f_i(y_t^i) \quad (1)$$

$$y_t^i = A_i x_t^i + b_i \quad (2)$$

$$x_t^i = z_t^{i-1} \quad \text{if } i > 0 \quad (3)$$

where z_t^i is the output of the i-th layer; f_i is the activation function of the i-th layer, say a sigmoid function for the hidden layers and a softmax function for the output layer; A_i and b_i are the weights of the i-th layer; x_t^i is the input to the i-th layer and z_t^{i-1} is the output of the $(i-1)$-th layer which serves as an input to the i-th layer.

For acoustic modeling, the DNN input often concatenates speech features with a context window of length L. Suppose o_t is the observation at time t, so $x_t^0 = [o_{t-L}, \ldots, o_t, \ldots, o_{t+L}]'$ is the input to the DNN. The output layer of the DNN is the state clustering output and z_t^{N+1} is the classification output of the DNN for x_t^0.

To train a DNN, cross entropy function is often used as an objective function, which computes the cross entropy between the probability distribution derived from the reference and the posterior probability distribution of the DNN output,

$$E = \sum_t \sum_j \gamma_t(j) \log z_{tj}^{N+1}\,, \tag{4}$$

where $\gamma_t(j)$ is the probability of being at state j at time t according to the reference (say, Viterbi alignment); z_{tj}^{N+1} is the posterior probability of being at state j at time t according to the DNN. The weights of the DNN (A_i and b_i) are often optimized by back-propagation which we differentiate the objective function with respect to A_i and b_i and perform gradient descent. Minimizing the cross entropy between γ and z means that the DNN matches the references better.

2.1. Linear Least Square Optimization for Neural Network

When a neural network contains no hidden layer, [16] proposes a training algorithm based on a linear least square (LLS) method. Assuming the activation function is invertible, one can derive the optimal y in the neural network, and optimize $W \equiv [A; b]$ by,

$$\begin{aligned} y_t^* &= f^{-1}(\gamma_t) && (5)\\ W^* &= \arg\min_W \frac{1}{2}\sum_t ||y_t^* - W'\tilde{x}_t||_2 && (6)\\ &= (\sum_t y_t^* x_t')(\sum_t \tilde{x}_t\tilde{x}_t')^+ && (7)\\ &\equiv Y_T X_T^+ && (8)\end{aligned}$$

where $\tilde{x}_t = [x_t', 1]'$ and X_T^+ is the pseudo-inverse of X_T. It is important to note that X_T is not full rank since $\tilde{x}$ is augmented by a constant 1.0. Using pseudo-inverse to solve this LLS problem means we are looking for a sparse solution, W^*, where its Frobenius norm is also minimum.

This LLS approach is optimal but it is limited to a neural network without hidden layer and the activation function of the output layer is invertible. To extend this approach for DNN with multiple hidden layers and non-invertible activation functions, say rectified linear units, we propose the DNN training can be decomposed into two steps. In the first step, we use gradient descent to optimize the internal representation of a DNN, i.e. y_t^i. Once the internal representation is optimized, we use LLS to optimize the DNN weights, i.e. $W_i \equiv [A_i, b_i]$, so that it is closest to the optimized internal representation. Using the cross entropy function as an example, we can compute

$$\begin{aligned} \frac{\partial E}{\partial y_t^i} &= \frac{\partial E}{\partial z_t^i}\frac{\partial z_t^i}{\partial y_t^i} && (9)\\ \hat{y}_t^i &= y_t^i - \lambda\frac{\partial E}{\partial y_t^i} && (10)\end{aligned}$$

so y_t^i can be optimized by back-propagation which is similar to using back-propagation to optimize the weights in standard DNN training. The target internal representation $\hat{y}_t^i$ is then approximated by,

$$\begin{aligned} W_i^* &= \arg\min_{W_i} \frac{1}{2}\sum_t ||\hat{y}_t^i - W'\tilde{x}_t^i||_2 && (11)\\ &= (\sum_t \hat{y}_t^i x_t^{i'})(\sum_t \tilde{x}_t^i\tilde{x}_t^{i'})^+ && (12)\\ &\equiv \hat{Y}_T^i(X_T^i)^+ && (13)\end{aligned}$$

For DNN with multiple hidden layers, one can first train the first layer. Then re-apply the procedure to train the next layer until it finishes training the entire DNN. Algorithm 1 describes the procedure for this optimization algorithm.

Algorithm 1 Our proposed LLS algorithm for DNN adaptation

```
for $i = 0, \ldots, N$ (for each layer) do
  for $t = 1, \ldots, T$ (for each frame) do
    compute $x_t^i, y_t^i, z_t^i$
    for $m = 1, \ldots, M$ do
      compute $x_t^j, y_t^j, z_t^j (\forall j > i)$ # partial forward
      compute $\frac{\partial E}{\partial y_t^i}$ # partial backward
      $y_t^i \leftarrow y_t^i - \lambda\frac{\partial E}{\partial y_t^i}$
      $z_t^i \leftarrow f_i(y_t^i)$
    end for
    $\hat{Y}_T^i \leftarrow \hat{Y}_T^i + \hat{y}_t^i x_t^{i'}$ # accumulate statistics
    $X_T^i \leftarrow X_T^i + \tilde{x}_t^i\tilde{x}_t^{i'}$
  end for
  $W_i^* \leftarrow \hat{Y}_T^i(X_T^i)^+$ # update the DNN weights
end for
```

Compared to the original LLS algorithm in [16], this proposed algorithm supports the use of hidden layers and it does not require the activation function to be invertible. This flexibility is at the expense of giving up the guarantee that the solution is optimal. Compared to the standard back-propagation algorithm for DNN training, this LLS method requires $O(N^2)$ of matrix operations which can be slow. However, as shown in our experiment in section 3, we found that it is often sufficient to train only the last layer of the DNN. Considering that this algorithm also processes data as a batch, it can be easily parallelized. Hence, the proposed algorithm is practical and efficient.

3. Experimental Results

3.1. System description and evaluation method

The IARPA BABEL program is a research program for rapid development of keyword spotting systems for low resource languages. In this work, we evaluate our proposed DNN adaptation algorithm using the BABEL development languages in year two, which include Assamese(babel102b-v0.5a), Bengali(babel103b-v0.4b), Haitian Creole(babel201b-v0.2b), Lao(IARPA-babel203b-v3.1a) and Zulu(babel206b-v0.1e). The evaluation has different conditions and one of them is the limited condition, in which the training consists of 10 hours of transcribed audio and roughly 90 hours of unsupervised data. The audio data is mainly conversational speech between two persons over a telephone channel, but each language pack also comes with a small amount of read speech. The development set for each language consists of roughly 10 hours of conversational telephone speech. The evaluation set, given by IARPA, contains 15 hours of speech for each language. The training and development data are labeled with environment types including public place, street, vehicle, car kit, home/office landline and home/office mobile phone. In which, Haitian and Zulu do not contain data for home/office landline. Dialect information for each audio is also provided. Table 1 is a summary of the dialects available for each language in the BABEL data.

Table 1: Dialects annotated for each language.

Language	Dialects
Assamese	Central Assam, Western Assam, Eastern Assam
Bengali	Varendra, Kamrupa, Radha
Haitian	Southern, Northern, Western
Lao	Vientiane
Zulu	Durban

For keyword search, each language has two set of keywords: a development keyword list and an evaluation keyword list. The development keyword list contains around 2000 keywords which were selected by the performers for development. The evaluation keyword lists consists of 3000 to 4000 keywords, and they were given during the evaluation. Each keyword may contain several words and they may or may not be in the training vocabulary. The performance of a keyword search system is measured by the Actual Term Weighted Value (ATWV) and WER is also measured for the underlying STT system. ATWV is computed by,

$$ATWV = 1 - \frac{1}{K}\sum_{w=1}^{K}\left(\frac{\#miss(w)}{\#ref(w)} + \beta\frac{\#fa(w)}{T - \#ref(w)}\right) \quad (14)$$

where K is the number of keywords; $\#miss(w)$ is the number of true keyword tokens that are not detected; $\#fa(w)$ is the number of false alarms; $\#ref(w)$ is the number of words in reference; T is the number of trials (e.g., seconds in the audio), and β is a constant set at 999.9. The details and the design of this metric are available in [18].

The BBN keyword search system is divided into several components. At a high level, the speech recognition system([19, 20]) is run to produce a detailed lattice of word hypotheses. This lattice is used to extract keyword hits with nominal posterior probability scores produced by various methods. In-vocabulary keywords are extracted using whole-word extraction and out-of-vocabulary keywords are extracted using phonetic extraction The scores are normalized so that they are consistent across keywords. Details of score normalization are available in [21].

Our DNN acoustic model consists of four hidden layers and each layer contains 2048 sigmoid units. We perform semi-supervised training as described in [22], in which, confidence weights are used for both cross entropy and sequence training. The features for the DNN acoustic model combine the regular PLP features with the semi-supervised MLP features provided by Brno University of Technology [23]. We use region dependent feature transformation [20] to project the combined features into 46 dimension and these features are further transformed by feature MLLR estimated using a GMM system. Finally, we concatenate the rotated features with a window size of 11 frames to produce the 506 dimension feature vectors as the DNN input.

3.2. Comparing Linear Least Square Adaptation and Fine-tuning

We compare our proposed LLS adaptation and fine-tuning for unsupervised speaker adaptation using the Bengali system. For adaptation, each speaker has around five minutes of audio and a GMM system is used to generate auto transcripts. For LLS adaptation, we compare adapting the entire DNN and just the output layer. For fine-tuning, we experiment different learning weights and we also apply silence weighting to improve adaptation performance.

Table 2: WER(%) of unsupervised speaker adaptation using LLS and fine-tuning on the Bengali system.

Adaptation	Setting	WER
Baseline	-	62.1
Fine-tuning	λ=0.001	62.4
Fine-tuning	λ=0.0005	61.9
Fine-tuning	λ=0.000125	61.9
Fine-tuning	λ=0.000125, sil_weight=0	63.6
Fine-tuning	λ=0.000125, sil_weight=0.1	62.0
Fine-tuning	λ=0.000125, sil_weight=0.25	61.7
Fine-tuning	λ=0.000125, sil_weight=0.5	61.7
LLS	λ=1.0, M=1, output layer	61.4
LLS	λ=1.0, M=1, entire DNN	61.2
LLS	λ=1.0, M=5, output layer	60.4
LLS	λ=1.0, M=5, entire DNN	60.4

Table 2 shows that by tuning the learning rate and applying a different weight to the silence frames, it is possible to achieve moderate improvement in WER from 62.1% to 61.7%. In contrast, LLS adaptation can improve the baseline system further (62.1% to 60.4%) and requires little tuning. As shown in this experiment, adapting the output layer gives about the same improvement compared to adapting the entire network. Computationally, performing five partial forward and backward operations ($M = 5$) costs roughly the same as the performing the full forward and backward computation once.

3.3. Unsupervised Environment, Dialect and Speaker Adaptation

Using the best configuration from the previous experiment, we test our algorithm for environment adaptation, dialect adaptation and speaker adaptation. For environment and dialect adaptation, the audio files are annotated with environment and dialect labels in the BABEL data, therefore, we randomly choose five minutes of the auto transcripts from the same environment or the same dialect from other audio files in the dev set to perform adaptation. The adaptation data for environment and dialect adaptation excludes the data from the same speaker in order to evaluate the effectiveness of such adaptation. For Lao and Zulu, the dialect adaptation results are omitted because there is only one dialect available in the BABEL data.

Table 3 shows the overall WER(%) for each language after performing unsupervised environment, dialect and speaker adaptation. In general, speaker adaptation achieves the best performance while environment and dialect adaptation still manages to outperform the baseline moderately. This is expected since the amount of adaptation data is roughly the same, speaker adaptation has the advantage of exploiting the information of the target speaker and channel. Figure 1 shows the improvement of LLS adaptation for each environment. We observe most en-

Table 3: WER(%) of unsupervised environment, dialect and speaker adaptation for each language.

Language	Baseline	Environment	Dialect	Speaker
Assamese	60.3	60.1	60.0	59.5
Bengali	62.1	61.3	61.4	60.4
Haitian	50.8	50.3	50.3	49.4
Lao	57.2	56.2	-	56.1
Zulu	68.4	67.5	-	67.6

vironment types have moderate improvement which shows our proposed algorithm is effective for environment adaptation. Table 4 shows the improvement for each dialect. While we observe improvement for Bengali, we do not see improvement for Assamese and Haitian. We believe dialect adaptation is generally harder which [14] shows it needs more than an acoustic model adaptation technique to achieve good performance.

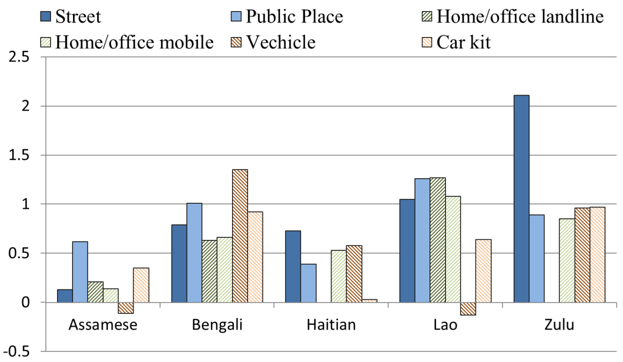

Figure 1: Absolute WER reduction of each environment with LLS adaptation compared to the unadapted baseline

Table 4: WER(%) of each dialect with and without LLS adaptation

Language	Assamese		
Dialect	Central	Eastern	Western
Baseline	60.5	58.4	61.3
Adapted	60.3	58.4	61.0
Language	Bengali		
Dialect	Varendra	Kamrupa	Radha
Baseline	61.0	64.7	61.0
Adapted	59.9	64.0	60.6
Language	Haitian		
Dialect	Northern	Southern	Western
Baseline	52.6	53.9	50.3
Adapted	52.6	53.7	50.2

3.4. Unsupervised Speaker Adaptation for Keyword Search

In this experiment, we explore how we can use the LLS algorithm to improve keyword search performance. As shown in our previous studies [22], improvement in WER does not necessarily improve keyword search performance. Our initial experiments on keyword search using LLS adaptation find that ATWV may degrade even if WER improves. For Bengali, LLS adaptation improves the WER from 62.1% to 60.4% but ATWV degrades from 38.4% to 37.1%. In our analysis, we find that many lattices become very shallow after LLS adaptation which suggests this adaptation basically reinforces the words in the auto transcripts and removes the less likely words in the lattices. This may hurt keyword search because some keywords may have low posterior probabilities in the lattices but later process in the pipeline may still be able to retrieve them. To handle this issue, we try to reduce the learning rate and decrease the number of iterations. In addition, instead of using the auto transcripts for adaptation, we also use the lattices produced by the GMM system to perform lattice based adaptation. The idea is instead of using the Viterbi alignment of the auto transcripts as references, we use the posterior probabilities in the lattices to represent the references. By doing so, LLS would not focus too much on a particular cluster target for each frame.

Table 5: ATWV(%) and WER(%) of LLS adaptation

	Assamese		Bengali		Haitian		Lao		Zulu	
	ATWV	WER	ATWV	WER	ATWV	WER	ATWV	WER	ATWV	WER
Baseline	34.6	60.3	38.4	62.1	50.9	50.8	47.0	57.2	24.9	68.4
M=5,λ=1.0	33.0	59.5	37.1	60.4	51.2	49.4	48.4	54.7	24.1	67.6
M=1,λ=0.25	34.8	60.1	38.7	61.6	52.2	50.2	47.7	56.1	26.3	67.3
+lat adapt	35.8	59.9	39.1	61.4	52.5	50.3	48.0	55.6	26.1	67.4

Table 5 shows that although less aggressive adaptation and lattice based adaptation may not give the best performance in WER, it improves keyword search performance by 0.7% to 1.6% absolute in ATWV for every language. It also maintains a moderate gain in WER (0.4% to 1.6% absolute) compared to the baseline results without LLS adaptation.

4. Conclusions and Future Work

In this paper, we propose a versatile unsupervised adaptation algorithm for DNN based on a linear least square method. We evaluate our approach on IARPA BABEL evaluation and show that this algorithm can improve both speech recognition and keyword search performance. We show that our proposed algorithm works even when the system has high error rate of 60-70%. Also, the type of adaptation is not constrained by the training data, so it is immediately applicable to improve any existing DNN acoustic model. In this paper, we investigate speaker, environment and dialect adaptation and we believe this algorithm also has the potential to adapt the DNN for unseen data (say unseen channel). In the future, we will explore speaker adaptive training for DNN and also rapid speaker adaptation which only uses a few seconds of adaptation data.

5. Acknowledgment

Supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Defense US Army Research Laboratory contract number W911NF-12-C-0013. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoD/ARL, or the U.S. Government.

6. References

[1] G. Hinton, L. Deng, D. Yu, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, G. Dahl, and B. Kingsbury, "Deep Neural Networks for Acoustic Modeling in Speech Recognition," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[2] G. 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, no. 1, pp. 30–42, 2012.

[3] B. Kingsbury, "Lattice-based Optimization of Sequence Classification Criteria for Neural Network Acoustic Modeling," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2009.

[4] K. Veselý, A. Ghoshal, L. Burget, and D. Povey, "Sequence-discriminative Training of Deep Neural Networks," in *Proceedings of the INTERSPEECH*, 2013.

[5] M. Karafiát, F. Grézl, M. Hannemann, K. Veselý, and J. H. Černocký, "BUT BABEL System for Spontaneous Cantonese," in *Proceedings of the INTERSPEECH*, 2013.

[6] K. Yao, D. Yu, F. Seide, H. Su, L. Deng, and Y. Gong, "Adaptation of Context-dependent Deep Neural Networks for Automatic Speech Recognition," in *IEEE Spoken Language Technology Workshop*, 2012, pp. 366–369.

[7] H. Liao, "Speaker Adaptation of Context Dependent Deep Neural Networks," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2013, pp. 7947 – 7951.

[8] D. Yu, K. Yao, H. Su, G. Li, and F. Seide, "KL-Divergence Regularized Deep Neural Network Adaptation For Improved Large Vocabulary Speech Recognition," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2013.

[9] G. Saon, H. Soltau, D. Nahamoo, and M. Picheny, "Speaker Adaptation of Neural Network Acoustic Models Using I-Vectors," in *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*, 2013.

[10] Y. Miao, H. Zhang, and F. Metze, "Towards Speaker Adaptive Training of Deep Neural Network Acoustic Models," in *Proceedings of the INTERSPEECH*, 2014.

[11] C. Leggetter and P. Woodland, "Maximum Likelihood Linear Regression for Speaker Adaptation of Continuous Density Hidden Markov Models," *Computer Speech and Language*, vol. 9, pp. 171–185, 1995.

[12] J. Gauvain and C. Lee, "Maximum a Posteriori Estimation for Multivariate Gaussian Mixture Observations of Markov Chains," *IEEE Transactions on Speech and Audio Processing*, vol. 2, no. 2, pp. 291–298, 1994.

[13] S. A. Yeung and M. Siu, "Improved Performance of Aurora 4 Using HTK and Unsupervised MLLR Adaptation," in *Proceedings of the International Conference on Spoken Language Processing*, 2004, pp. 221–224.

[14] T. Schultz and A. Waibel, "Language Independent and Language Adaptive Acoustic Modeling for Speech Recognition," *Speech Communication*, vol. 35, no. 1–2, pp. 31–51, 2001.

[15] P. Karanasou, Y. Wang, M. Gales, and P. Woodland, "Adaptation of Deep Neural Network Acoustic Models Using Factorised I-Vectors," in *Proceedings of the INTERSPEECH*, 2014.

[16] E. Castillo, O. Fontenla-Romero, B. Guijarro-Berdiñas, and A. Alonso-Betanzos, "A Global Optimum Approach for One-layer Neural Networks," *Neural Computation*, vol. 14, no. 6, pp. 1429–1449, 2002.

[17] D. Yu and L. Deng, "Efficient and Effective Algorithms for Training Single-Hidden-Layer Neural Networks," *Pattern Recognition Letters*, vol. 3, no. 5, pp. 554–558, 2012.

[18] "OpenKWS13 Keyword Search Evaluation Plan," http://www.nist.gov/itl/iad/mig/upload/OpenKWS13-EvalPlan.pdf, 2013.

[19] R. Hsiao, T. Ng, F. Grézl, D. Karakos, S. Tsakalidis, L. Nguyen, and R. Schwartz, "Discriminative Semi-supervised Training for Keyword Search in Low Resource Languages," in *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*, 2013.

[20] T. Ng, B. Zhang, S. Matsoukas, and L. Nguyen, "Region Dependent Transform on MLP Features for Speech Recognition," in *Proceedings of the INTERSPEECH*, 2011.

[21] D. Karakos, R. Schwartz, S. Tsakalidis, L. Zhang, S. Ranjan, T. Ng, R. Hsiao, G. Saikumar, I. Bulyko, L. Nguyen, J. Makhoul, F. Grézl, M. Hannemann, M. Karafiát, I. Szoke, K. Veselý, L. Lamel, and V.-B. Le, "Score Normalization and System Combination for Improved Keyword Spotting in Speech," in *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*, 2013.

[22] R. Hsiao, T. Ng, L. Zhang, S. Ranjan, S. Tsakalidis, L. Nguyen, and R. Schwartz, "Improving Semi-supervised Deep Neural Network for Keyword Search in Low Resource Languages," in *Proceedings of the INTERSPEECH*, 2014.

[23] F. Grézl and M. Karafiát, "Semi-Supervised Bootstrapping Approach for Neural Network Feature Extractor Training," in *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*, 2013.