Speaker adaptation of neural network acoustic models using i-vectors

Conference Paper · December 2013

DOI: 10.1109/ASRU.2013.6707705

4 authors, including:

George Saon
IBM
102 PUBLICATIONS **2,123** CITATIONS
SEE PROFILE

Michael Picheny
IBM
154 PUBLICATIONS **3,628** CITATIONS
SEE PROFILE

Speaker Adaptation of Neural Network Acoustic Models Using I-Vectors

George Saon, Hagen Soltau, David Nahamoo and Michael Picheny

IBM T. J. Watson Research Center, Yorktown Heights, NY, 10598

***Abstract*—We propose to adapt deep neural network (DNN) acoustic models to a target speaker by supplying speaker identity vectors (i-vectors) as input features to the network in parallel with the regular acoustic features for ASR. For both training and test, the i-vector for a given speaker is concatenated to every frame belonging to that speaker and changes across different speakers. Experimental results on a Switchboard 300 hours corpus show that DNNs trained on speaker independent features and i-vectors achieve a 10% relative improvement in word error rate (WER) over networks trained on speaker independent features only. These networks are comparable in performance to DNNs trained on speaker-adapted features (with VTLN and FMLLR) with the advantage that only one decoding pass is needed. Furthermore, networks trained on speaker-adapted features and i-vectors achieve a 5-6% relative improvement in WER after hessian-free sequence training over networks trained on speaker-adapted features only.**

I. Introduction

Given the recent popularity of deep neural networks for acoustic modeling, speaker adaptation of DNNs is an active area of research [1], [2], [3], [4], [5]. However, the portability of transform-based approaches like MLLR that work well for Gaussian mixture models to DNNs is not straightforward. Unlike Gaussian means or variances which can be transformed together if they belong to the same acoustic class (phones, HMM states or clustered versions thereof), it is hard to find structure in the weights of a neural network . Rather, researchers have looked at approaches analogous to MAP for GMMs where the weights of the network are updated directly using the adaptation data of a given speaker. The problem with this approach is that the number of parameters that are updated far exceeds the amount of adaptation data available which can lead to overfitting and some form of regularization is necessary [5]. Alternatively, [2] have looked at adapting only the biases. Another approach suggested in [1] is to add a linear layer between the frames and the input layer that can be trained similar to FMLLR (although with a cross-entropy criterion instead of ML).

Yet another approach that is getting some traction in the literature is to sidestep the network adaptation problem altogether and train networks on speaker-adapted features instead. Such features can be extracted using the speaker normalization machinery readily available for GMM-HMMs such as vocal tract length normalization and feature-space MLLR. This approach works well despite the fact that the VTLN and FMLLR transforms are estimated assuming a GMM-HMM acoustic model and are now being used in conjunction with a DNN-HMM.

A better way might be to provide the network with untransformed features and let it figure out the speaker normalization during training. In order to do that, the network has to be informed which features belong to which speaker. This can be accomplished by creating two sets of time-synchronous inputs: one set of acoustic features for phonetic discrimination and another set of features that characterize the speaker which provided the audio for the first set of features. This idea is similar to [3] with one important difference: in our proposed work, the features which characterize a speaker are the same for all the data of that speaker. Another work relevant to ours is [4], where the authors propose to learn speaker codes which are fed to a speaker adaptation network. The network produces speaker-adapted features which form the input to a regular DNN. The main difference in our proposed work (besides using i-vectors instead of speaker codes) is that we train a single network that does speaker adaptation and phone classification simultaneously instead of two separate networks. Lastly, noise-aware DNNs proposed in [6], which use as input uncompensated features and time-dependent estimates of the noise, are also relevant to our work.

Why speaker recognition features should be helpful can be shown through a simple thought experiment. Imagine that there are two types of speakers, say A and B, which differ in the way they pronounce the phone /AA/. Speaker type A uses the canonical pronunciaton /AA/ whereas speaker type B systematically pronounces it as /AE/. A DNN without speaker features will tend to classify B's /AA/ as /AE/ because statistically there will be more /AE/'s with canonical pronunciations in the training data. A DNN with speaker identity features however, will learn to significantly increase the output score for /AA/ when presented with /AE/ acoustics for speakers of type B (but not for speakers of type A). In other words, the network can learn speaker-dependent transforms for the acoustic features in order to create a canonical phone classification space in which inter-speaker variability is significantly reduced.

I-vectors [7] are a popular technique for speaker verification and speaker recognition because they encapsulate all the relevant information about a speaker's identity in a low-dimensional fixed-length representation [8]. This makes them an attractive tool for speaker adaptation techniques for ASR. A concatenation of i-vectors and ASR features is used in [9] for discriminative speaker adaptation with region dependent linear transforms. I-vectors are also employed in [10], [11] for clustering speakers or utterances on mobile devices for more efficient adaptation. The attractiveness of i-vectors and

these previous works (notably [9]) motivated us to look at their applicability to speaker adaptation of DNNs for ASR.

The paper is organized as follows: in section II we review the i-vector extraction method, in section III we provide some experimental results for DNNs trained with and without i-vectors on the Switchboard English conversational telephone task, and in section IV we summarize our findings.

II. I-VECTOR TECHNIQUE

Here we describe the main ideas behind the i-vector technique. Allthough an exhaustive treatment of i-vectors can be found in many works (see, for example, [8] and the references therein), we outline the main points here in order for the paper to be self-contained.

Borrowing some notations from [10], the acoustic feature vectors $\mathbf{x}_t \in \mathbb{R}^D$ are seen as samples generated from a universal background model (or UBM) represented as a GMM with K diagonal covariance Gaussians

$$\mathbf{x}_t \sim \sum_{k=1}^{K} c_k \mathcal{N}(\cdot; \boldsymbol{\mu}_k(0), \boldsymbol{\Sigma}_k) \tag{1}$$

with mixture coefficients c_k, means $\boldsymbol{\mu}_k(0)$ and diagonal covariances $\boldsymbol{\Sigma}_k$. Moreover, data $\mathbf{x}_t(s)$ belonging to speaker s are drawn from the distribution

$$\mathbf{x}_t(s) \sim \sum_{k=1}^{K} c_k \mathcal{N}(\cdot; \boldsymbol{\mu}_k(s), \boldsymbol{\Sigma}_k) \tag{2}$$

where $\boldsymbol{\mu}_k(s)$ are the means of the GMM adapted to speaker s. The essence of the i-vector algorithm is to assume a linear dependence between the speaker-adapted means $\boldsymbol{\mu}_k(s)$ and the speaker-independent means $\boldsymbol{\mu}_k(0)$ of the form

$$\boldsymbol{\mu}_k(s) = \boldsymbol{\mu}_k(0) + \mathbf{T}_k \mathbf{w}(s), \qquad k = 1 \dots K \tag{3}$$

$\mathbf{T}_k$, of size $D \times M$, is called the factor loading submatrix corresponding to component k and $\mathbf{w}(s)$ is the speaker identity vector ("i-vector") corresponding to s. Each $\mathbf{T}_k$ contains M bases which span the subspace with important variability in the component mean vector space. The two questions that need to be answered are: (i) given $\mathbf{T}_k$ and speaker data $\{\mathbf{x}_t(s)\}$ how do we estimate $\mathbf{w}(s)$? and (ii) given training data $\{\mathbf{x}_t\}$ how do we estimate the matrices $\mathbf{T}_k$?

A. I-vector estimation

From a bayesian perspective, $\mathbf{w}$ is treated as a latent variable with a 0-mean, identity covariance Gaussian prior distribution and we estimate the posterior distribution of $\mathbf{w}$ given speaker data $\{\mathbf{x}_t(s)\}$, i.e. $p(\mathbf{w}|\{\mathbf{x}_t(s)\})$. Under the assumption of a fixed (soft) alignment of frames to mixture components, it can be shown that this posterior distribution is Gaussian [12]

$$p(\mathbf{w}|\{\mathbf{x}_t(s)\}) = \mathcal{N}(\mathbf{w}; \mathbf{L}^{-1}(s) \sum_{k=1}^{K} \mathbf{T}_k^T \boldsymbol{\Sigma}_k^{-1} \boldsymbol{\theta}_k(s), \mathbf{L}^{-1}(s)) \tag{4}$$

with precision matrix $\mathbf{L}(s)$ of size $M \times M$ expressed as

$$\mathbf{L}(s) = \mathbf{I} + \sum_{k=1}^{K} \gamma_k(s) \mathbf{T}_k^T \boldsymbol{\Sigma}_k^{-1} \mathbf{T}_k \tag{5}$$

The quantities that appear in (4) and (5) are the zero-order and centered first-order statistics and are defined as

$$\gamma_k(s) = \sum_t \gamma_{tk}(s), \tag{6}$$

$$\boldsymbol{\theta}_k(s) = \sum_t \gamma_{tk}(s)(\mathbf{x}_t(s) - \boldsymbol{\mu}_k(0)) \tag{7}$$

with $\gamma_{tk}(s)$ being the posterior probability of mixture component k given $\mathbf{x}_t(s)$. The i-vector that we are looking for is simply the MAP point-estimate of the variable $\mathbf{w}$ which is the mean of the posterior distribution from (4), i.e.

$$\mathbf{w}(s) = \mathbf{L}^{-1}(s) \sum_{k=1}^{K} \mathbf{T}_k^T \boldsymbol{\Sigma}_k^{-1} \boldsymbol{\theta}_k(s) \tag{8}$$

B. Factor loading matrix estimation

Model hyperparameters $\{\mathbf{T}_1, \dots, \mathbf{T}_K\}$ are estimated using the EM algorithm to maximize the ML objective function [13]

$$Q(\mathbf{T}_1, \dots, \mathbf{T}_K) = -\frac{1}{2} \sum_{s,t,k} \gamma_{tk}(s) \Big[\log |\mathbf{L}(s)| + (\mathbf{x}_t(s) - \boldsymbol{\mu}_k(s))^T \boldsymbol{\Sigma}_k^{-1} (\mathbf{x}_t(s) - \boldsymbol{\mu}_k(s)) \Big] \tag{9}$$

which can be written equivalently as

$$Q(\mathbf{T}_1, \dots, \mathbf{T}_K) = -\frac{1}{2} \sum_{s,k} \Big[\gamma_k(s) \log |\mathbf{L}(s)| + \gamma_k(s) Tr\{\boldsymbol{\Sigma}_k^{-1} \mathbf{T}_k \mathbf{w}(s) \mathbf{w}(s)^T \mathbf{T}_k^T\} - 2 Tr\{\boldsymbol{\Sigma}_k^{-1} \mathbf{T}_k \mathbf{w}(s) \boldsymbol{\theta}_k(s)^T\} \Big] + \mathcal{C} \tag{10}$$

The term $\log |\mathbf{L}(s)|$ comes from the logarithm of the posterior $p(\mathbf{w}|\{\mathbf{x}_t(s)\})$ evaluated in $\mathbf{w}(s)$. Taking the derivative of (10) with respect to $\mathbf{T}_k$ and setting it to 0 leads to collecting the sufficient statistics [8]

$$\mathbf{C}_k = \sum_s \boldsymbol{\theta}_k(s) \mathbf{w}^T(s), \tag{11}$$

$$\mathbf{A}_k = \sum_s \gamma_k(s) (\mathbf{L}^{-1}(s) + \mathbf{w}(s) \mathbf{w}^T(s)) \tag{12}$$

where $\mathbf{L}^{-1}(s)$ and $\mathbf{w}(s)$ are given respectively by (5) and (8) for speaker s. The factor loading submatrices are updated as follows

$$\mathbf{T}_k = \mathbf{C}_k \mathbf{A}_k^{-1}, \qquad k = 1 \dots K \tag{13}$$

In summary, the i-vector extraction transforms are estimated iteratively by alternating between the E-step (11),(12) and the M-step (13).

C. Integration with a DNN

As shown in Figure 1, the procedure for using i-vectors with a neural network is as follows. First, the speaker data $\{\mathbf{x}_t(s)\}$ is aligned with the GMM to estimate the zero-order and first-order statistics from (6) and (7). These quantities are then used to estimate the i-vector $\mathbf{w}(s)$ via (5) and (8). Next, $\mathbf{w}(s)$ is concatenated to every frame $\mathbf{x}_t(s)$ to form the input for neural network training or decoding.

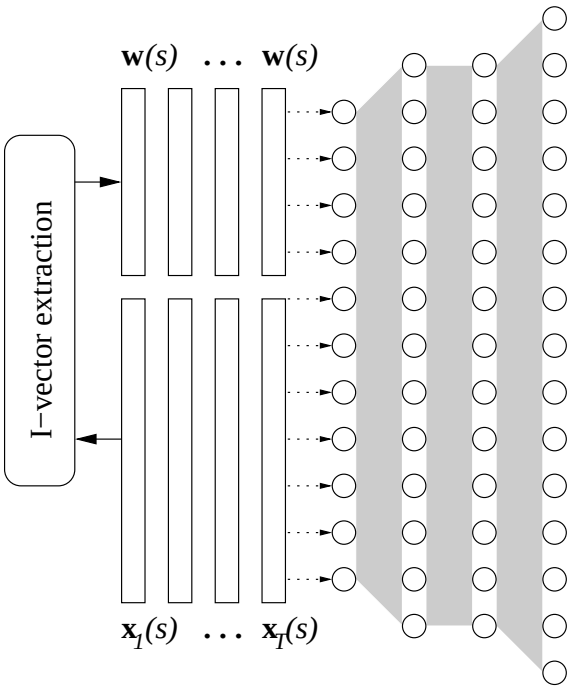

Fig. 1. I-vector extraction and input features for a neural network.

III. Experiments and Results

Following [1], we conducted our experiments on a 300 hour subset of the Switchboard English conversational telephone speech task. We report results on the testsets that were used during the Hub5 2000 and Rich Transcription 2003 Darpa evaluations which will be referred to as the Hub5'00 and RT'03 evaluation sets. These testsets contain 2.1 hours of audio, 21.4K words and 7.2 hours of audio, 76K words, respectively.

A. Frontend processing

Speech is coded into 25 ms frames, with a frame-shift of 10 ms. Each frame is represented by a feature vector of 13 perceptual linear prediction (PLP) cepstral coefficients which are mean and variance normalized per conversation side. Every 9 consecutive cepstral frames are spliced together and projected down to 40 dimensions using LDA. The range of this transformation is further diagonalized by means of a global semi-tied covariance transform. Additionally, for the speaker-adapted features, the cepstra are warped with vocal tract length normalization (VTLN) prior to splicing and projection. Then, one feature-space MLLR (FMLLR) transform per conversation side is computed on top of the LDA features at both training and test time using a GMM-HMM system.

B. I-vector extraction

We use the maximum likelihood criteria to train two 2048 40-dimensional diagonal covariance GMMs: one for the speaker-independent and one for the speaker-adapted feature sets. These GMMs were used to precompute the zero and first-order statistics via (6) and (7) for all the training and test speakers. The i-vector extraction matrices $\mathbf{T}_1, \ldots, \mathbf{T}_{2048}$ were initialized with values drawn randomly from the uniform distribution in $[-1, 1]$ and were estimated with 10 iterations of EM by alternating the sufficient statistics collection (11),(12) and the factor subloading matrix update (13). Once the matrices were trained, we extracted M-dimensional i-vectors for all the training and test speakers. This procedure was repeated for 3 different values of M: 40, 100 and 200. Lastly, the i-vectors were scaled so that they have approximately unit variance on the training data for neural network training.

C. DNN training

Several networks were trained which differ in the type of input features: speaker-independent (SI) and speaker-adapted (SA) and in whether they have i-vector input or not. All networks share the following characteristics. The input features use a temporal context of 11 frames as suggested in [1] meaning that the input layer has either $40 \times 11 + M$ (for $M \in \{40, 100, 200\}$) or 40×11 neurons for nets with and without i-vector inputs. The training data is divided randomly at the speaker level into a 295 hours training set and a 5 hours held-out set.

All nets have 6 hidden layers with sigmoid activation functions: the first 5 with 2048 units and the last one with 256 units for parameter reduction and faster training time [14]. The output layer has 9300 softmax units that correspond to the context-dependent HMM states obtained by growing a phonetic decision tree with pentaphone crossword context.

Following the recipe outlined in [1], the training data is fully randomized at the frame level within a window of 25 hours and we trained the nets with stochastic gradient descent on minibatches of 250 frames and a cross-entropy criterion. Prior to the cross-entropy training of the full network, we used layerwise discriminative pretraining by running one cross-entropy sweep over the training data for the intermediate networks obtained by adding one hidden layer at a time. Additionally, we applied hessian-free sequence training for some of the networks using a state-based minimum Bayes risk objective function as described in [15].

D. Hybrid DNN-HMM decoding

The trained DNNs are used directly in a hybrid decoding scenario by subtracting the logarithm of the HMM state priors

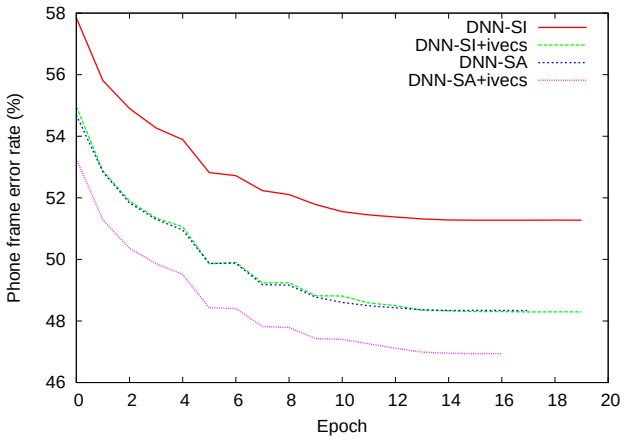

Fig. 2. Phone frame error rates on heldout data for various DNNs.

from the log of the DNN output scores. The vocabulary used has 30.5K words and 32.8K pronunciation variants. The decoding language model is a 4-gram LM with 4M n-grams.

E. Experimental results

In Figure 2, we compare the phone frame error rates obtained on the held-out set during the cross-entropy fine-tuning (i.e. after pretraining) of 4 networks: a DNN on SI features only, a DNN on SI features and i-vectors of dimension 100, a DNN on SA features only and a DNN on SA features and i-vectors of dimension 100. We observe that DNNs with i-vector inputs are substantially better than the ones trained on ASR features only. Interestingly, the curve for DNNs trained on SI features and i-vectors is almost indistinguishable from the one obtained by DNNs trained on SA features only which suggests that the i-vector input has the same effect as adding VTLN and FMLLR.

Model	Training	Hub5'00	RT'03	
		SWB	FSH	SWB
DNN-SI	x-entropy	16.1%	18.9%	29.0%
DNN-SI	sequence	14.1%	16.9%	26.5%
DNN-SI+ivecs	x-entropy	13.9%	16.7%	25.8%
DNN-SI+ivecs	sequence	12.4%	15.0%	24.0%
DNN-SA	x-entropy	14.1%	16.6%	25.2%
DNN-SA	sequence	12.5%	15.1%	23.7%
DNN-SA+ivecs	x-entropy	13.2%	15.5%	23.7%
DNN-SA+ivecs	sequence	11.9%	14.1%	22.3%

TABLE I
COMPARISON OF WORD ERROR RATES FOR VARIOUS DNNS ON HUB5'00 AND RT'03 WITHOUT AND WITH HESSIAN-FREE SEQUENCE TRAINING.

This is also mirrored in the word error rates shown in Table I where the DNN-SI+ivecs and DNN-SA models exhibit very similar recognition performance (10% relative WER improvement over DNN-SI). Additionally, we observe that DNN-SA with i-vectors results in a 5-6% relative improvement over DNN-SA both before and after sequence training. The additive gains can be explained by observing that the i-vectors for DNN-SA were extracted using a GMM trained on speaker-adapted features as opposed to using a UBM trained on speaker independent features for DNN-SI. This allows the i-vectors to encode additional salient speaker information after the VTLN and FMLLR speaker normalization steps.

i-vector dimension	Training	Hub5'00 SWB	RT'03 FSH	RT'03 SWB
40	x-entropy	13.7%	16.0%	24.6%
100	x-entropy	13.2%	15.5%	23.7%
200	x-entropy	13.4%	15.6%	23.6%

TABLE II
COMPARISON OF WORD ERROR RATES FOR DNNS TRAINED ON SPEAKER-ADAPTED FEATURES AND I-VECTORS OF DIFFERENT DIMENSIONS ON HUB5'00 AND RT'03 WITH CROSS-ENTROPY TRAINING.

Lastly, we discuss the effect of having different i-vector dimensions in Table II for DNNs trained on speaker-adapted features with cross-entropy only (no sequence training). It can be seen that having a sufficiently large dimension for the i-vectors matters; there is a significant drop in WER from 40 to 100 dimensions. The performance is flat for 100 and 200 dimensions which suggests that having an i-vector dimension of 100 is a reasonable choice for this task. Of course, more training data from a larger number of speakers might result in a different operating point.

IV. CONCLUSION

We have presented a simple yet effective way to perform speaker adaptation for neural network acoustic models. The method consists in providing speaker identity vectors alongside regular ASR features as inputs to the neural net. The training and test data are augmented with these i-vectors which are constant for a given speaker and change across different speakers. Unlike other speaker adaptation techniques, i-vector extraction does not require a first pass decoding step yet provides similar gains as VTLN and FMLLR which do require an additional decoding pass. Moreover, i-vectors extracted from speaker-adapted features are complementary to the feature normalization methods applied and provide additional gains when used in conjunction with speaker normalized features as input to the neural networks. Future work will address refining the i-vector inputs along the lines of the speaker code idea proposed in [4]. Also, we plan to interleave neural network training with speaker identity feature estimation analogous to speaker-adaptive training in feature-space for GMM-HMMs.

ACKNOWLEDGMENT

The authors wish to thank Jason Pelecanos from IBM and Ondrej Glembek from Brno University of Technology for helpful discussions about i-vectors.

REFERENCES

[1] F. Seide, G. Li, X. Chien, and D. Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription," in *Proc. ASRU*, 2011.

[2] 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 *Proc. SLT*, 2012.

[3] M. Ferras and H. Bourlard, "MLP-based factor analysis for tandem speech recognition," in *Proc. of ICASSP*, 2013.

[4] O. Abdel-Hamid and H. Jiang, "Fast speaker adaptation of hybrid NN/HMM model for speech recognition based on discriminative learning of speaker code," in *Proc. of ICASSP*, 2013.

[5] 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 *Proc. of ICASSP*, 2013.

[6] M. Seltzer, D. Yu, and Y. Wang, "An investigation of deep neural networks for noise robust speech recognition," in *Proc. of ICASSP*, 2013.

[7] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Trans. Audio, Speech and Language Processing*, vol. 19, no. 4, May 2011.

[8] O. Glembek, L. Burget, P. Matejka, M. Karafiat, and P. Kenny, "Simplification and optimization of i-vector extraction," in *Proc. ICASSP*, 2011.

[9] M. Karafiat, L. Burget, P. Matejka, O. Glembek, and J. Cernozky, "iVector-based discriminative adaptation for automatic speech recognition," in *Proc. ASRU*, 2011.

[10] K. Yao, Y. Gong, and C. Liu, "A feature space transformation method for personalization using generalized i-vector clustering," in *Proc. Interspeech*, 2012.

[11] M. Bacchiani, "Rapid adaptation for mobile speech applications," in *Proc. of ICASSP*, 2013.

[12] P. Kenny, "Joint factor analysis of speaker and session variability: theory and algorithms," CRIM Technical Report, Tech. Rep., 2006.

[13] N. Brummer, "The EM algorithm and minimum divergence," Agnitio Labs Technical Report, Tech. Rep., 2009.

[14] T. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, "Low-rank matrix factorization for deep neural network training with high-dimensional output targets," in *Proc. of ICASSP*, 2013.

[15] B. Kingsbury, T. Sainath, and H. Soltau, "Scalable minimum Bayes risk training of deep neural network acoustic models using distributed Hessian-free optimization," in *Proc. Interspeech*, 2012.