IMPROVING DNN SPEAKER INDEPENDENCE WITH I-VECTOR INPUTS

Andrew Senior, Ignacio Lopez-Moreno

Google Inc.,
New York
{andrewsenior,elnota}@google.com

ABSTRACT

We propose providing additional utterance-level features as inputs to a deep neural network (DNN) to facilitate speaker, channel and background normalization. Modifications of the basic algorithm are developed which result in significant reductions in word error rates (WERs). The algorithms are shown to combine well with speaker adaptation by backpropagation, resulting in a 9% relative WER reduction. We address implementation of the algorithm for a streaming task.

***Index Terms*—** Deep neural networks, large vocabulary speech recognition, Voice Search, i-vectors, speaker adaptation.

1. INTRODUCTION

Deep neural networks have come to prominence as acoustic models in recent years, surpassing the performance of the previous dominant paradigm, Gaussian Mixture Models (GMMs). One of the most powerful techniques for improving the accuracy of GMM speech models has been speaker adaptation wherein a speaker independent model is adapted on a small amount of data from a single speaker, with the resulting speaker-specific model performing better on test data from that speaker. Several studies [1, 2, 3] have shown that speaker adaptation is less effective with DNNs than with GMM acoustic models, partly because of the greater invariance of DNNs to speaker variations and their higher baseline accuracy.

Nevertheless, these studies do show that deep networks can be made more invariant to speaker variability. One of the problems with speaker adaptation is that it is hard to adapt a large number of parameters with only a small amount of data. Care must be taken to change the parameters sufficiently to have an effect without overfitting on the new data. Further, speaker adaptation results in a new model, or part-model, for each speaker which, in a cloud-based speech recognizer adds significant complexity and storage.

1.1. Deep networks

Recent results by many groups [4] have shown significant accuracy improvements over GMMs by using DNNs either to generate the GMM features or to directly estimate the acoustic model scores. Neural networks consist of many simple units which each compute a weighted sum of the activations of other units, and output an activation which is a nonlinear function of that sum. Typically these units are arranged in layers which receive input from the units in the previous layer, with the first layer computing a weighted sum of externally provided features, such as the filterbank energies of frames of speech. These networks can be trained to approximate a desired output function by the backpropagation of the error in the output compared to a target value provided for each training input example. We have previously applied hybrid DNNs for acoustic modelling in Google's VoiceSearch [5, 6] and YouTube [7] applications.

1.2. Speaker adaptation (of DNNs)

The classic techniques for speaker adaptation of Gaussian Mixture Models are (Constrained) Maximum Likelihood Linear Regression (CMLLR) [8, 9]) and Maximum A Posteriori modelling [10]. In the former, a linear transformation, computed to maximize the likelihood of the adaptation data, is applied to the features. This technique has been applied to the features input to a neural network, but has the limitation of requiring the transform to be computed with a GMM which also limits the dimensionality and types of features which can be used. We have found that the gains from using high dimensional, stacked mel scale log filterbank energies over using conventional low-dimensional speech features outweigh the gains from being able to do CMLLR adaptation. Bacchiani [11] has shown that GMMs can be speaker-adaptated using utterance i-vectors (Section 2).

Abrash *et al.* [2] showed that neural networks can be adapted by training an input transform or adapting the whole network with backpropagation, and Liao [3] has recently shown that these techniques can be applied to DNNs with millions of parameters, although the gains are smaller on larger networks which are inherently more speaker-independent than smaller networks.

Ström [12] showed that a neural network system trained with speaker identities could be used at inference time without knowing the speaker's identity, inferring a *speaker space vector* and reducing the WER by 2.5% relative. Abdel-Hamid and Jiang [13, 14] recently proposed providing speaker adaptation in a DNN by learning a similar speaker code which is used to compute speaker-normalized features. In experiments on the TIMIT dataset, they used backpropagation to learn a separate code for each speaker. This speaker code was then used as an input to the network for utterances by the same speaker. These experiments showed 5% relative phone error rate reductions with DNNs.

Seltzer *et al.* [15] have shown that augmenting the inputs of a neural network with an estimate of background noise level can improve the robustness of such a network to background noise. This "noise-aware" training gave a 4% relative improvement compared to a DNN baseline using the dropout technique.

While this paper was under review, Saon *et al.* published a study [16] in which they augment DNN inputs with *speaker* i-vector features, whereas we use *utterance* i-vectors in a similar manner. They demonstrate a 10% relative reduction in WER on the 300 hour Switchboard task.

2. I-VECTORS

In the speaker recognition community utterances are typically represented by a *supervector*, whose components are the Maximum A Posteriori (MAP) adaptation coefficients of a large Gaussian Mixture Model (GMM) known as the Universal Background Model (UBM).

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

A number of factors such as the speaker identity and so-called session factors can contribute to the variability in the parameters N and F. Session factors include undesired variation associated with the utterance length, phonetic dependency and environmental conditions. In the last few years Factor Analysis (FA) has proved to be successful in modelling these components of variability as low dimensional *latent* variables (*i.e.* manifolds).

Several alternative FA methods have been used for speaker recognition, namely Joint Factor Analysis (JFA) [17], Total Variability (TV) [18] and more recently, Probabilistic Linear Discriminant Analysis (PLDA) [19]. Unlike JFA, where the undesired session variability and the useful speaker variability are explicitly modelled as two non-overlapping manifolds, the TV model has shown superior performance by modelling all sources of variability in the supervector as a single manifold. A point in this space of latent variables is referred as an "identity vector", or i-vector. The PLDA model can be seen as a combination of the previous two techniques, focused on extracting the speaker variability from the utterance i-vector.

Since they provide a compact representation of speaker and session factors that we wish a speech recognition system to be invariant to, i-vectors and other FA-based factors have been used in the past for rapid speaker adaptation of speech recognition systems. However, most of these contributions were based on classical HMM-based acoustic models. The Eigenvoices model [20] uses short-term HMM-derived speaker factors (i.e. eigenvoices) to bring a general speech recognition model closer to a particular speaker, and Bacchiani [11] used i-vectors for a better modelling of session variability, demonstrating an 11% WER reduction..

2.1. Computing i-vectors

Utterance supervectors are typically represented by the accumulated and centered zero- and first-order Baum-Welch statistics, N and F respectively. N and F statistics are computed from a UBM, denoted by λ. For UBM mixture $m \in 1, \ldots, C$, with mean, μ_m, the corresponding zero- and centered first-order statistics are aggregated over all frames in the database:

$$N_m = \sum_t P(m|o_t, \lambda), \quad (1)$$

$$F_m = \sum_t P(m|o_t, \lambda)(o_t - \mu_m), \quad (2)$$

where $P(m|o_t, \lambda)$ is the Gaussian occupation probability for the mixture m given the spectral feature observation $o_t \in \Re^D$ at time t. The TV model can be seen as a classical FA generative model [21], with observed variables given by the vector of stacked statistics $F = \{F_1, F_2, \ldots, F_m\}$. The TV model defines a set of hidden variables $x \in \Re^L : P(x) = \mathcal{N}(0, I)$ and a Gaussian distribution $P(x|F)$ that represents the utterance. In order to formulate $P(x|F)$, the model imposes a Gaussian distribution over $P(F|x)$, which relates observed and hidden variables in terms of a the rectangular low rank matrix $T \in \Re^{CD \times L}$:

$$P(F \mid x) = \mathcal{N}(NTx, \Sigma), \quad (3)$$

being $\Sigma \in \Re^{CD \times CD}$ a diagonal covariance matrix in the space of F. Here, N denotes a diagonal matrix of size $CD \times CD$ formed by C diagonal blocks of size $D \times D$ where the m-th component block is given the matrix $N_m I_{(D \times D)}$.

The utterance i-vector is defined as the value of x that maximizes $P(x|F)$ -the mean value-. For the imposed values of $P(x)$ and $P(F|x)$ the i-vector is formulated as:

$$x = (I + T^t \Sigma^{-1} N T)^{-1} T^t \Sigma^{-1} F, \quad (4)$$

Size	Context		Layers	Units per layer	Output states	Params
	L	R				
Small	10	5	4	480	1000	1.5M
Medium	10	5	6	512	2000	2.7M
Large	16	5	6	2176	14247	70M

Table 1: Parameters for the fully-connected sigmoid neural networks with softmax outputs.

The TV model is thus a data driven model with parameters $\{\lambda, T, \Sigma\}$. In [18] the authors provide a more detailed explanation of deriving these parameters, using the EM algorithm.

3. ADAPTING DNNS WITH I-VECTORS

Here we propose the idea that i-vectors can be used as input features for neural networks, resulting in improved recognition. i-vectors encode precisely those effects to which we want our ASR system to be invariant: speaker, channel and background noise. While the targets to which we normally train are independent of these factors, providing the network with a characterisation of them at the input should enable it to normalise the signal with respect to them and thus better able to make its outputs invariant to them.

Consequently, we propose augmenting the traditional acoustic input features with the utterance i-vector. A network which takes a context window of c frames of d dimensional acoustic features is augmented with v i-vector dimensions resulting in a $cd + v$ dimensional input, as shown in Figure 1.

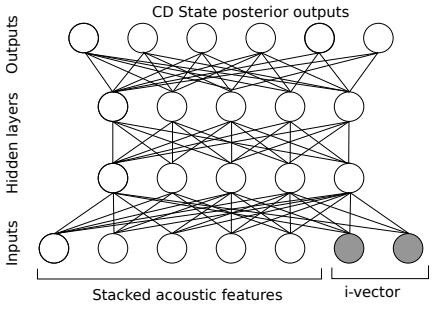

Fig. 1: Diagram of a 2-hidden layer neural network with inputs augmented with i-vectors.

As with traditional cross-entropy training, frames from the training data are randomly selected and stacked with the appropriate context window but all frames from a given utterance are augmented with the same v dimensional utterance i-vector.

3.1. Baseline Experiments

In our first experiments we trained three different sizes of network, with and without utterance i-vectors. The network configurations were chosen to suit both "cloud" speech recognition on a conventional server as well as two sizes of "embedded" speech recognizers designed to run on mobile phones of different processing power. Each network is fully connected with logistic sigmoid hidden layers and softmax outputs, receiving stacked 25ms frames of 40-dimensional Mel filterbank energy features as input. The number of parameters in the baseline networks are shown in Table 1, with the augmented networks having slightly more parameters in the initial layer because of the increased input dimension. All the networks

are trained from random initialization with exponentially decaying learning rates.

The networks are trained on a corpus of 3 million utterances (about 1,750 hours) of US English Google voice search and dictation traffic, anonymized and hand-transcribed. This data is endpointed and aligned using a high accuracy server-sized neural network with 14247 context dependent (CD) states. For the smaller networks these state symbols are mapped through equivalence classes down to the smaller state inventories. During training, CD state frame accuracies are evaluated on the training data and on a held out development set of 200,000 frames. Word Error Rates (WERs) are measured on a test set of 23,000 hand-transcribed utterances sampled from live traffic. Training is by stochastic gradient descent with a minibatch size of 200 frames on a Graphics Processing Unit.

The parameters of the TV model, including the UBM, were also trained on this corpus. The UBM was trained with 1024 mixtures computed from 13 perceptual linear prediction coefficients with delta and delta-delta features appended. The matrix Σ was built by stacking the diagonal covariance matrices and never updated, while the matrix T was initialized using PCA and updated with 10 EM iterations for 300 latent variables.

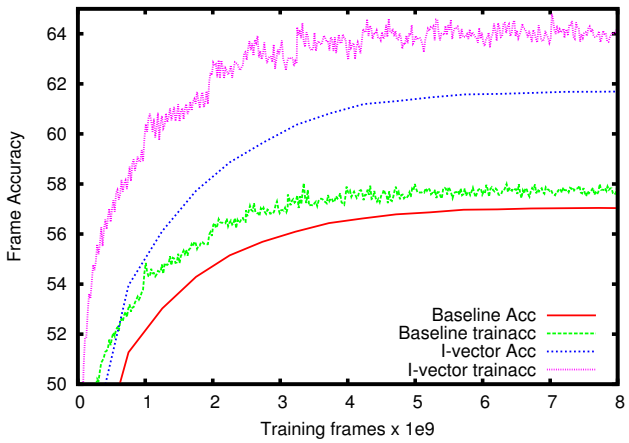

Fig. 2: Frame accuracies against billions of training samples on training and held-out dev sets during training for the larger network, with and without i-vector inputs.

Model size	Baseline		i-vector	
	WER	Frame acc.	WER	Frame acc.
Small	**17.8**	55.4	18.2	58.5
Medium	**15.0**	55.0	15.5	59.1
Large	**11.0**	57.1	12.3	59.1

Table 2: WERs and development set frame accuracy for baseline and with inputs augmented by 300 i-vector dimensions.

Figure 2 shows the progress of development set and training set frame accuracies during training of the small networks, and Table 2 shows the corresponding WERs. Early in training the i-vectors give a significant (3%) increase in development set frame accuracy and a larger (6%) increase in training set frame accuracy. As training progresses, the margin between training and dev-set accuracy diminishes, but the margin for the i-vector-augmented network remains much larger than for the baseline (2% vs 0.8%). From these graphs we infer that the network is able to use the i-vector to predict the frame classes, but is overfitting to the i-vector and is unable to use this information during decoding, resulting in higher WERs. Alternatively, it is possible that many voice search utterances are very short -only a few hundreds of voiced frames- and the 300-dimensional i-vectors estimate is not reliable. Both conjectures will explored in the next section.

4. REGULARIZATION

To avoid the overfitting we investigated two solutions:

1. Reduce the information content of the i-vectors.
2. Regularize the network parameters.

The first solution attempts to reduce the overfitting by limiting the amount of information presented to the network. To this end we truncate the i-vectors to a smaller dimension before augmenting the input vector. Arbitrarily the first k elements of the vector are preserved. Table 3 shows that reducing the dimensionality of the i-vector results in a lower WER, with the greatest gains being made with the smaller networks. Having validated the use of

Model size	i-vector dimensions (k)					
	0	20	50	100	200	300
Small	17.8	**17.0**	17.2	17.4	17.9	18.2
Medium	15.0	**14.5**	**14.5**	**14.5**	15.2	15.5
Large	11.0	**10.9**	**10.9**	11.2	11.8	12.3

Table 3: WERs when augmenting the network inputs with truncated i-vector inputs.

lower dimensional i-vectors, we trained low dimensional TV matricess with 20 and 50 dimensions and repeated the above experiment with these i-vectors. We found that training a medium network with a 20-dimensional i-vector led to a WER of 14.4%, outperforming the 300-dimensional i-vector truncated to 20 dimensions, but a 50-dimensional i-vector performing worse, at 14.9% WER.

The second solution begins with a network trained without any i-vector information. This network's input layer's weight matrix is augmented with weights, initially set to zero, from 300 additional inputs. The network is then trained further with i-vector-augmented inputs, but with ℓ_2 regularization (weight decay) back to the original weights, *i.e.* adding a term to the loss function proportional to the sum-squared difference between the network's weights and those of the network before i-vector augmentation. Experimentation found good results with a weight decay parameter of 10^{-7} to 10^{-6}. The small, medium and large networks were augmented, before full convergence, at 12, 10 and 4 billion frames respectively. Table 4 shows

Training	Original	Regularization	
		10^{-7}	10^{-6}
Small	17.8	**17.2**	17.3
Medium	15.0	**14.5**	14.6
Large	11.0	**10.6**	10.8

Table 4: WERs when training an i-vector-augmented network while regularizing back to the original weights.

the results of this regularization. We see that the regularized networks have a lower WER than the original, unaugmented features, and that the large regularized network outperforms the corresponding network trained with truncated i-vectors.

5. COMBINING WITH ADAPTATION

In this section we explore the interaction between using utterance i-vectors for invariance and the use of adaptation to provide speaker invariance. Liao [3] describes how training on an adaptation set for a particular speaker using backpropagation with ℓ_2 regularization back to the original, speaker-independent, model can reduce WERs on test data from the same speaker.

Here we compare our technique with this approach and show that the two can be used in combination to achieve even lower error rates. These experiments are conducted with the best "Medium" models only. We use the same personalization training and test sets used by Liao. These have 10 minutes of adaptation data for each of 80 speakers, and a total of 10,000 utterances (72,000 words) from the same 80 speakers in the test set. We report average word error rates across the entire test set. We use the "enrollment" protocol, i.e. the training data is manually transcribed and force-aligned with a large DNN model. The baseline model is adapted to each speaker's data with multiple passes. We continue to use the same, exponentially decaying learning rate, and an ℓ_2 regularization weight of 0.01. We find the best performance after 1 million frames of adaptation.

The results are shown in Table 5. As can be seen, the adapted baseline model achieves a lower word error rate than with the unadapted i-vector-augmented models, but when the latter are also adapted, their WER is also reduced, bringing a total of 9% relative WER reduction from the combined technique for the truncated i-vectors.

Model	Unadapted	Adapted
Baseline model	15.3	14.4
300 dim Regularized (10^{-7}) model	14.9	14.3
20-dim i-vector model	**14.7**	**14.0**

Table 5: WERs for medium DNNs on the personalization test set representing 80 speakers, using speaker independent models and models adapted on 10 minutes of data per speaker (with 1 million frames of adaptation).

6. STREAMING IMPLEMENTATION

One limitation of this approach is that the i-vector representation we are using is computed on an entire utterance, and thus can only be computed when all the data for an utterance is available. Our principal application is real-time transcription of utterances from mobile devices with minimum latency, which involves processing utterances as they are being spoken and streaming results back to the user even before the utterance is complete. This approach produces a very responsive speech interaction on the device, but means that whole-utterance approaches cannot be used in practice, although fast rescoring with the augmented models could be applied without introducing too much overall latency.

To address this incompatibility with our application we investigated using speaker-averaged i-vectors in place of utterance i-vectors. Here we compute an averaged speaker i-vector on the 10-minute adaptation set and used this for decoding the speaker's personalization data using the models trained above on per-utterance i-vectors. As can be seen in Table 6, the mismatched average i-vectors are not useful in decoding on a model trained with utterance i-vectors. In addition to matched training with speaker i-vectors, there are a number of alternative ways of training with ivectors for a streaming application which we will investigate in the future.

Model	Utterance	Speaker
Baseline model	15.3	
300 dim Regularized (10^{-7}) model	14.9	15.3
20-dim i-vector model	14.7	15.5

Table 6: WERs for medium DNNs on the 80 speaker personalization test set using speaker and utterance i-vectors in decoding.

- **On-line computation of the i-vectors:** We can compute the i-vectors based on the data so far, or use the *d-Vector* proposed by Variani et al. [22] computed, like our posteriors, based on a sliding window of frames.
- **Use of the i-vector from the speaker's previous utterance** Within a session, we expect variations in background noise, channel and speaker to be small, so the i-vector of the previous utterance may still be sufficient to provide invariance to these factors.

7. COMPARISON TO SIMILAR WORK

As noted earlier, Saon *et al.* published a similar work [16] while this paper was under review. They also augment the DNN input, but use the speaker i-vector for all utterances by the speaker, both in training and testing. Their DNNs are trained on LDA-projected PLP features from a narrow window which allow the use of conventional speaker adaptation. They show better performance with higher dimensional speaker i-vectors and obtain 10% relative WER reduction over speaker-independent features — their greater gains perhaps being due to the poorer baseline features used. They also demonstrated that i-vector augmentation combined well with a conventional speaker adaptation technique (CMLLR). They found that the i-vector dimension had to be at least 100, and in addition found that this technique was was beneficial in combination with sequence-discriminative training.

8. CONCLUSIONS

We have shown that using the utterance i-vectors as input features provides the neural networks with valuable information that, with the regularization we propose, bring about roughly a 4% relative reduction in word error rate for all model sizes. These techniques can be applied on any utterance, without requiring any speaker information or speaker adaptation or model storage. The technique has been shown to combine well with model adaptation, delivering an overall 9% WER reduction for models that are small enough to be run in real-time in a smart-phone, which are ideal candidates for speaker-adapted models.

These improvements are directly applicable to non-realtime applications, but are not well suited to a streaming scenario. We have proposed a variety of methods to address this in future work, but using the speaker i-vector in place of the utterance i-vector at test time did not help. It will be instructive to further investigate the relative benefit of using speaker i-vectors compared to utterance i-vectors which are far more noisy (particularly on short utterances) but offer independence to variations other than speaker identity.

9. ACKNOWLDEGEMENTS

Thanks to Hank Liao for providing the Personalization data sets and testing protocols.

10. REFERENCES

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

[2] V. Abrash, H. Franco, A. Sankar, and M. Cohen, "Connectionist speaker normalization and adaptation," in *Proc. Eurospeech*, 1995.

[3] H. Liao, "Speaker adaptation of context dependent deep neural networks," in *Proc. ICASSP*, 2013.

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

[5] N. Jaitly, P. Nguyen, A. W. Senior, and V. Vanhoucke, "Application of pretrained deep neural networks to large vocabulary speech recognition," in *Proc. Interspeech*, 2012.

[6] X. Lei, A. Senior, A. Gruenstein, and J. Sorenson, "Accurate and compact large vocabulary speech recognition on mobile devices," in *Proc. Interspeech*, 2013.

[7] H. Liao, E. McDermott, and A.W. Senior, "Large scale deep neural network acoustic modeling with semi-supervised training data for YouTube video transcription," in *Proc. ASRU*, 2013.

[8] M.J.F. Gales, "Maximum likelihood linear transformations for HMM-based speech recognition," *Computer Speech and Language*, vol. 12, 1998.

[9] C.J. Leggetter and P.C. Woodland, "Maximum likelihood linear regression speaker adaptation of contiuous density HMMs," in *Computer Speech and Languages*, 1997.

[10] J.L. Gauvain and C.H. Lee, "Bayesian learning of Gaussian mixture densities for hidden Markov models," in *Proc. DARPA Speech and Natural Language Workshop*, 1991.

[11] M. Bacchiani, "Rapid adaptation for mobile speech applications," in *Proc. ICASSP*, 2013, pp. 7903–7907.

[12] N. Ström, "Speaker adaptation by modeling the speaker variation in a continuous speech recognition system," in *Proc. ICSLP*, 1996.

[13] 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. ICASSP*, 2013.

[14] O. Abdel-Hamid and H. Jiang, "Rapid and effective speaker adaptation of convolutional neural network based models for speech recognition," in *Proc. Interspeech*, 2013.

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

[16] G. Saon, H. Soltau, D. Nahamoo, and M. Picheny, "Speaker adaptation of neural network acoustic models using I-vectors," in *Proc. ASRU*, 2013.

[17] P. Kenny, "Joint Factor Analysis of Speaker and Session Variability: Theory and Algorithms," Available from: http://www.crim.ca/perso/patrick.kenny/FAtheory.pdf, in preparation.

[18] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-End Factor Analysis for Speaker Verification," *IEEE Trans. on Audio, Speech and Language Processing*, vol. 19, no. 4, pp. 788 – 798, February 2011.

[19] P. Kenny, "Bayesian speaker verification with heavy-tailed priors," in *Proc. Odyssey Speaker and Language Recognition Workshop*, 2010.

[20] R. Kuhn, J.-C. Junqua, P. Nguyen, and N. Niedzielski, "Rapid speaker adaptation in eigenvoice space," *IEEE Trans. on Audio, Speech and Language Processing*, vol. 8, no. 6, pp. 695–707, 2000.

[21] C.M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*, Springer, 1st ed. 2006. corr. 2nd printing edition, Oct. 2007.

[22] E. Variani, X. Lei, E. McDermott, and I. Lopez-Moreno, "Text-dependent speaker verification using deep neural networks," in *Proc. ICASSP*, 2014.