# SEQUENCE SUMMARIZING NEURAL NETWORK FOR SPEAKER ADAPTATION

*Karel Veselý*[†], *Shinji Watanabe*[§], *Kateřina Žmolíková*[†], *Martin Karafiát*[†]
*Lukáš Burget*[†], *Jan "Honza" Černocký*[†]
[†] Brno University of Technology, Brno, Czech Republic
[§]Mitsubishi Electric Research Laboratories (MERL), Cambridge, USA
{iveselyk,karafiat,burget,cernocky}@fit.vutbr.cz, watanabe@merl.com

## ABSTRACT

In this paper, we propose a DNN adaptation technique, where the i-vector extractor is replaced by a Sequence Summarizing Neural Network (SSNN). Similarly to i-vector extractor, the SSNN produces a "summary vector", representing an acoustic summary of an utterance. Such vector is then appended to the input of main network, while both networks are trained together optimizing single loss function. Both the i-vector and SSNN speaker adaptation methods are compared on AMI meeting data. The results show comparable performance of both techniques on FBANK system with frame-classification training. Moreover, appending both the i-vector and "summary vector" to the FBANK features leads to additional improvement comparable to the performance of FMLLR adapted DNN system.

***Index Terms*—** DNN, adaptation, i-vector, sequence summary, SSNN

## 1. INTRODUCTION

The adaptation methods for Deep Neural Networks (DNNs) are a very active area of research. The traditional approach which leads to good results is training a DNN on speaker adapted FMLLR features [1], which introduces multi-pass decoding and a dependency on a GMM model.

To avoid the dependency, one can apply the unsupervised self-training of the DNN. While using multi-pass decoding, a small amount of speaker-specific parameters inside the DNN is trained using the 1st pass hypotheses [2, 3].

Other popular methods use embedding of auxiliary information to the DNN input, which avoids the multi-pass decoding. Very popular are i-vectors [4, 5], which encode both the speaker and channel characteristics in a compact fixed-length representation.

Soon after the i-vectors revolutionized the speaker identification systems, they influenced the Automatic Speech Recognition (ASR). At first as additional input features to discriminatively trained Region Dependent Linear Transform for GMMs [6], later as additional input features of a DNN [7, 8, 9], while the i-vectors were also successfully used in robust ASR [10, 11, 12].

An alternative paradigm of using i-vectors as extra input features is to train a small adaptation network, which converts the i-vectors into offsets of input-features of the main network [13]. In this way, we can easily return to the speaker independent model by producing no offsets if we do not have enough data to estimate the i-vectors properly.

Yet another approach to extract a fixed length speaker representation is to average the activations in the last hidden layer, which generates the "d-vectors". This can be done in a feed-forward NN [14], or in LSTM [15]. In both cases, the networks are trained for per-frame classification of speakers. The averaging is done after the neural network is trained, which is fundamentally different from our approach. In our approach, the "sequence summary" representation is trained together with the main network, directly optimizing its objective function.

In our work, we first used i-vectors as extra input features, then we replaced the i-vector extractor by a *Sequence Summarizing Neural Network (SSNN)*. Similarly to i-vector extractor, the SSNN produces a "summary vector", which represents acoustic summary of an utterance. The "summary vector" is obtained by enclosing a sequence-averaging operation into the last component of the SSNN. The "summary vector" is then appended to the input of main network, and both networks are trained together, while the gradients for SSNN are calculated by back-propagating through the main network, which is trained to optimize a single loss function. The experiments show that comparable WER reduction can be achieved both by i-vector and "summary vector" approach on FBANK system with frame-classification training.

In section 2 we develop the idea of neural-network based "summary vector", while in section 3 we provide formal formulation of the back-propagation through the necessary sequence-averaging operation. Then in section 4 we describe the experimental setup, while section 5 presents the results, followed by a conclusion.

## 2. "SUMMARY VECTOR", NN-BASED ALTERNATIVE OF I-VECTOR

In the literature we have seen that using i-vectors as extra input features leads to WER reductions [7, 8, 9], while at the same time we introduce a dependency on the i-vector extractor and its models (GMM-UBM, T-matrix, Voice Activity Detection). To estimate the i-vector properly, we also need to have enough data for the target speaker.

In this paper, we propose an alternative, which removes this dependency. We generate a neural-network based "summary vector" similar to i-vector. As illustrated in Figure 1, this can be done by connecting two neural networks, where the output of the *Sequence Summarizing Neural Network (SSNN)* is averaged over a sentence and concatenated to the input of the main network. This average vector represents a summary of the sentence, while its dimension is independent on the length of the sequence.

The work reported here was carried out during the 2015 Jelinek Memorial Summer Workshop on Speech and Language Technologies at the University of Washington, Seattle, and was supported by Johns Hopkins University via NSF Grant No IIS 1005411, and gifts from Google, Microsoft Research, Amazon, Mitsubishi Electric, and MERL. BUT authors were also supported by Technology Agency of the Czech Republic project No. TA04011311 "MINT" and Czech MoI project No. VI20152020025 "DRAPAK".

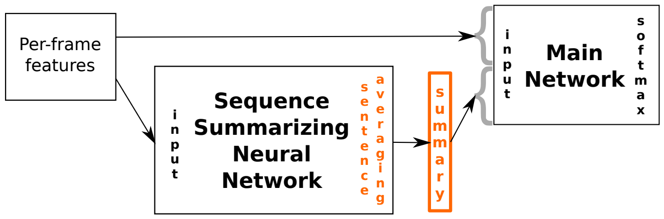

**Fig. 1**. *Topology of main-network with "sequence summary" input. The summary is computed by Sequence Summarizing Neural Network with sentence-averaging on the output.*

## 3. FORMULATION

This section provides the formulation of feed-forward and back-propagation of the proposed architecture of the two neural networks : auxiliary SSNN network and the main network. Recall that the key point is that the output of auxiliary network is averaged over a sentence and used as input of the main network.

### 3.1. Feed-forward

Let $\mathbf{o}_u = \{\mathbf{o}_{u,t} \in \mathbb{R}^D | t = 1, \ldots, T_u\}$ be a sequence of $D$ dimensional feature vector of utterance $u$. The output posterior probability $y_{u,t}(s)$ of the DNN for the HMM state $s$ is obtained using the softmax activation function:

$$y_{u,t}(s) \triangleq p(s|\mathbf{o}_{u,t}) = \frac{\exp(a_s(\mathbf{o}_{u,t}; \theta_o)))}{\sum_{s'} \exp(a_{s'}(\mathbf{o}_{u,t}; \theta_o))}, \tag{1}$$

where $a_s(\cdot)$ is the activation at the output layer for state $s$ given $\mathbf{o}_{u,t}$ with a set of parameters $\theta_o$. The set $\theta_o$ includes the affine transformation parameters for all the DNN layers.

In our framework, the original input feature vector $\mathbf{o}_{u,t}$ is *augmented* by an $K$ dimensional vector $\mathbf{x}_u \in \mathbb{R}^K$ to form the following $D + K$ dimensional feature:

$$\bar{\mathbf{o}}_{u,t} \triangleq \begin{bmatrix} \mathbf{o}_{u,t} \\ \mathbf{x}_u \end{bmatrix}. \tag{2}$$

Note that the vector $\mathbf{x}_u$ augmenting the utterance $u$ is independent on time, and it can involve utterance-level characteristics of speech (e.g., speaker, speaking style, room acoustics, and stationary noise). For example when i-vectors are used as extra features in the observation vector, they are usually constant within an utterance.

In our proposed framework, the augmented vector is obtained by the average of output vectors of an auxiliary network with output $\mathbf{x}_{u,t}$, which is represented as follows:

$$\mathbf{x}_u(\mathbf{o}_u; \theta_x) \triangleq \frac{1}{T_u} \sum_{t=1}^{T_u} \mathbf{x}_{u,t}(\mathbf{o}_{u,t}; \theta_x), \tag{3}$$

where $\theta_x$ is a set of parameters of the auxiliary network to be learned from data, and $\mathbf{o}_u \triangleq \{\mathbf{o}_{u,1}, \cdots, \mathbf{o}_{u,T_u}\}$. Then, in Eq. (1), we can replace the activation function $a_s(\cdot)$ with the following new function $b_s(\cdot)$:

$$a_s(\mathbf{o}_{u,t}; \theta_o) \rightarrow b_s(\mathbf{o}_{u,t}, \mathbf{x}_u(\mathbf{o}_u; \theta_x); \bar{\theta}_o). \tag{4}$$

Thus, the new activation function is represented by the auxiliary network parameters $\theta_x$ and the parameters $\bar{\theta}_o$, which are the original parameters $\theta_o$ extended by connections for new input features. The next section describes the back-propagation of the proposed network, which jointly learns both $\bar{\theta}_o$ and $\theta_x$.

### 3.2. Back-propagation

The back propagation of the extended original parameters $\bar{\theta}_o$ is the same as the conventional one. The objective function $\mathcal{F}$ can be either frame cross-entropy or sequence-discriminative criteria (e.g., MMI, sMBR [16]). The derivative is represented as:

$$\frac{\partial \mathcal{F}}{\partial \bar{\theta}_o} = \frac{\partial \mathcal{F}}{\partial b_s(\mathbf{o}_{u,t}, \cdot)} \frac{\partial b_s(\mathbf{o}_{u,t}, \cdot)}{\partial \bar{\theta}_o}. \tag{5}$$

where for simplicity we use $b_s(\mathbf{o}_{u,t}, \cdot)$ for the activations instead of $b_s(\mathbf{o}_{u,t}, \mathbf{x}_u(\mathbf{o}_u; \theta_x); \bar{\theta}_o)$. Note that the gradient can be computed separately for each observation frame $\mathbf{o}_{u,t}$, especially when we use the frame cross-entropy criterion as $\mathcal{F}$.

Now, we focus on the solution of the back propagation of $\theta_x$. By using the chain rule, the derivative of $\mathcal{F}$ with respect to $\theta_x$ is represented as follows:

$$\frac{\partial \mathcal{F}}{\partial \theta_x} = \frac{\partial \mathcal{F}}{\partial b_s(\mathbf{o}_{u,t}, \cdot)} \frac{\partial b_s(\mathbf{o}_{u,t}, \cdot)}{\partial \mathbf{x}_u(\mathbf{o}_u; \theta_x)} \frac{\partial \mathbf{x}_u(\mathbf{o}_u; \theta_x)}{\partial \theta_x}. \tag{6}$$

We use the chain rule to substitute the derivative of Eq. (3) into Eq. (6), while we consider that the loss for all frames in the sequence (i.e. utterance) depends on $\mathbf{x}_u(\mathbf{o}_u; \theta_x)$. The derivative is then rewritten as:

$$\frac{\partial \mathcal{F}}{\partial \theta_x} = \left[ \frac{1}{T_u} \sum_{t'=1}^{T_u} \frac{\partial \mathcal{F}}{\partial b_s(\mathbf{o}_{u,t'}, \cdot)} \frac{\partial b_s(\mathbf{o}_{u,t'}, \cdot)}{\partial \mathbf{x}_u(\mathbf{o}_u; \theta_x)} \right] \frac{\partial \mathbf{x}_{u,t}(\mathbf{o}_{u,t}; \theta_x)}{\partial \theta_x}. \tag{7}$$

Thus, we can derive the back-propagation of $\theta_x$ analytically. Note that the derivative of the sequence-averaging operation has analogical function form to the original averaging function. The error derivative is averaged over the whole utterance $u$ with length $T_u$, which was not the case in Eq. (5). This is a unique property of the proposed network allowing the utterance-level characteristics to be learned (or compensated) by the network.

## 4. EXPERIMENTAL SETUP

The experiments were performed on AMI meeting conversation corpus[1], using the Independent Headset Microphone (IHM) recordings. The data were recorded with 16 kHz sampling rate, the train-set contains 77 hours of data, while the dev and eval sets consist of 9 hours each.

For the DNN experiments, we needed to build *initial GMM-HMM models* to produce the tied-state clustering, alignments and FMLLR features. The GMM-HMM models are trained on features obtained by splicing together 7 frames (3 on each side of the current frame) of 13-dimensional MFCCs (C0-C12) and projecting down to 40 dimensions using linear discriminant analysis (LDA). The MFCCs are normalized to have zero mean per speaker. We also use a single semi-tied covariance (STC) transform on the features obtained using LDA. Moreover, speaker adaptive training (SAT) is done using a single feature-space maximum likelihood linear regression (FMLLR) transform estimated per speaker.

All our speech recognition systems are HMM-based with cross-word tied-states triphones. The initial GMM-HMM system was trained using standard mixing-up maximum likelihood training. The DNNs models were built using the following two recipes: a simplified one and a complete one.

[1]`http://corpus.amiproject.org/`

### 4.1. Simplified DNN recipe

The *simplified recipe* consists of frame-classification training of randomly initialized DNN with 4 hidden layers composed of 1024 sigmoid neurons each, and softmax output layer with 3977 neurons.

The input weight matrix was initialized from $\mathcal{N}(0, 0.001)$, the 3 following initial weight matrices had samples from $\mathcal{N}(0, 0.01)$ and the output layer weight matrix was initialized from $\mathcal{N}(0, 0.005)$. The different variances ensure that the gradients in the 1st mini-batch will be "reasonable", which was discussed in [17]. The hidden biases were initialized from $\mathcal{U}(-4.0, 0)$, while the initial output biases were set to zero.

The input features are 40 log Mel-filterbank outputs extended by 3 Kaldi-pitch features [18]. We apply per-speaker mean and variance normalization, frame splicing of 21 frames (10 on each side of current frame). Each temporal trajectory is scaled by Hamming window and reduced by Discrete Cosine Transform with 11 basis. The final 473-dimensional features are globally shifted and scaled to have zero mean and unit variance on the DNN input.

### 4.2. Complete DNN recipe

The *complete recipe* consists of RBM-pretraining, frame-classification training and the sequence-discriminative training. The network has 6 hidden layers of 2048 sigmoid neurons each and a softmax output layer with 3977 outputs.

The DNNs were trained on the LDA+STC+FMLLR features obtained from the initial GMM-HMM system, here we applied splice of 11 frames (5 on each side of current frame) and global normalization to have zero mean and unit variance in the 440 dimensional DNN input.

The DNN was initialized with Restricted Boltzmann Machine (RBM) pre-training [19], using the setup described in [20]. After pre-training, we appended randomly initialized softmax output layer and continued with frame-classification training which minimizes the cross-entropy between the triphone tied-state posteriors and the Viterbi alignment. The last stage of the recipe are 4 epochs of sequence-discriminative training which optimizes Sequence Minimum Bayes Risk (sMBR) objective [16], while the learning-rate is fixed to $10^{-5}$.

### 4.3. Common for both DNN recipes

For the i-vector systems, we used *mini-batch SGD* training with initial learning-rate 0.008 and 256-frame mini-batches. For systems with "sequence summary" we used *SGD with per-utterance updates* and initial learning-rate 0.004.

In all the per-frame classification trainings, we used learning-rate scheduling based on relative improvement of the frame cross-entropy on 10% held-out set. We start to halve the learning-rate when the relative improvement falls below 0.01, and the training ends if the relative improvement is lower than 0.001, which is usually after 14 epochs.

To be able to use RBM pre-training with "summary vector" on input, we first trained a small network to initialize the Sequence Summarizing Neural Network.

### 4.4. I-vector extractor

We used the i-vector extractor from the BUT standardization initiative[2], it is trained on 9000 hour corpus composed of Fisher English (part 1 and 2), NIST SRE 2004-2008, Switchboard (phase 2, phase 3, cellular part 1, and cellular part 2). The i-vectors have 600 dimensions, we generated one i-vector vector per speaker, while using a multi-lingual NN-based VAD tuned to detect confident speech. The produced i-vector dataset was both mean and length-normalized, while we did not use Within Class Covariance Normalization (WCCN). We also compared 600-dimensional and 100-dimensional i-vectors, and observed no difference in WER performance.

[2]`http://voicebiometry.org/`

### 4.5. Sequence Summarization Neural Network (SSNN)

The SSNN is trained jointly with the main network using the standard back-propagation algorithm, as discussed in section 3. We successfully trained the architecture of the two networks from random initialization, while using the per-utterance update SGD.

The SSNN consists of 2 Tanh layers with 512 neurons and a 600-dimensional layer with linear output, where the linear outputs are averaged over the entire sequence. The biases were initialized to 0, while the weight matrices were initialized from $\mathcal{N}(0, 0.0036)$, here the variance was tuned manually. The learning rate for SSNN is the same as for the main network.

In case of *simplified recipe*, the insertion of SSNN to the baseline DNN caused the increase of number of parameters from 7.7 to 9.1 millions. In the *complete recipe* the increase was from 30.1 to 32.0 millions of parameters.

## 5. RESULTS

The first experiment compares the mini-batch training and the training with per-utterance updates. As we see in Table 1, the insignificant difference of 0.1% shows that both are equally good for our database.

All the results in Table 1 are obtained with the "simplified DNN recipe". The simplification is that the topology is smaller, there is no RBM pre-training and sMBR training, and the input features are speaker independent. By comparing the 1st and 2nd block in Table 1, we see that the use of i-vectors or the "summary vectors" leads to similar performance improvements on eval set: 1.6% for i-vector, 1.4% for "summary vector", while on the dev set the 0.8% i-vector improvement is better than 0.5% "summary vector" improvement, compared to the unadapted baselines.

In the 3rd block of Table 1, we used the speaker-adapted FMLLR features instead of the speaker-independent FBANK features. Here, the performance improvement from the "summary vector" method is smaller: 0.1% on dev set and 0.4% on eval set. If we consider i-vectors, "summary vectors" and FMLLR as alternative methods to choose from, the eval set suggests that FMLLR is the best, while the differences on dev set are smaller. However, we should also consider practical limitations: both the i-vector and FMLLR methods need several utterances for each speaker, while for the "summary vector" all we need is a single utterance.

We also tried to use both the "summary vectors" and i-vectors as the extra input features of the main network. The result in the 2nd block in Table 1 shows complementary behavior reaching the performance of the FMLLR system without any extra features (1st row in 3rd block).

We also experimented with the per-sentence i-vectors extracted from the frames where VAD detected speech. However, the performance was 4% worse than the baseline. This could be possibly fixed by keeping a context of few previous sentences as done for the train-

**Table 1**. *Comparing the i-vector and "summary vector" speaker adaptation of DNN on "simplified recipe": 4 layers, random initialization, FBANK features with Hamming-DCT processing.*

| Simplified DNN recipe | WER% | |
|---|---|---|
| | dev | eval |
| mini-batch training | 27.9 | 31.4 |
| + i-vector | 27.1 | **29.8** |
| per-utterance update | 28.0 | 31.3 |
| + "summary vector" | 27.5 | **29.9** |
| + "summary vector" and i-ivector | 27.0 | 29.2 |
| per-utterance update (FMLLR) | 27.4 | 28.8 |
| + "summary vector" | 27.3 | 28.4 |

**Table 2**. *Comparing the i-vector and "summary vector" speaker adaptation of DNN on very challenging "complete recipe": 6 layers, RBM pre-training, FMLLR features.*

| Complete DNN recipe | WER% | |
|---|---|---|
| | dev | eval |
| Baseline | | |
| - frame training | 25.8 | 27.1 |
| - sMBR | 24.2 | 24.7 |
| i-vector | | |
| - frame training | 25.6 | 26.3 |
| - sMBR | **23.8** | **23.9** |
| "summary vector" | | |
| - frame training | 26.0 | 26.6 |
| (no RBM, random init.) | 26.4 | 27.0 |
| - sMBR | 24.5 | 24.6 |

ing of i-vector extractor in [9], but this already deviates from the per-sentence processing we use for SSNN.

The results for the "complete DNN recipe" are in Table 2, where we again compare the i-vector and "summary vector" systems with the baseline. Here the baseline has speaker-adapted FMLLR features, a large topology initialized by RBM pre-training, and trained by per-frame training and sequence-discriminative sMBR training. After sMBR training, the i-vector system outperforms *the very challenging* baseline system by 0.4% on dev set and 0.8% on eval set. On the other hand, the comparison of sMBR "summary vector" system with the sMBR baseline system shows a little degradation of -0.3% on dev-set and tiny improvement of 0.1% on eval-set. This result is a little disappointing, however if we subdivide the dev set by sentence lengths as shown in figure 2, we see that the system with "summary vectors" outperforms both the baseline and i-vector DNN systems on sentences longer than 10 seconds, which is promising. Certainly there is an open space for further investigation.

Last, as the RBM pre-training with "summary vectors" on input requires SSNN which is already trained, we tried to replace the RBM initialization with random initialization. This leads to performance degradation of 0.4% on both test sets (3rd block of Table 2).

## 6. CONCLUSIONS AND DISCUSSION

In this paper, we proposed an alternative method to produce DNN adaptation vectors similar to i-vectors. The vectors are computed by the Sequence Summarizing Neural Network and characterize the acoustics in an utterance. This is done by enclosing sentence-averaging operation in the last component. The SSNN network is trained together with the main network using standard back-propagation algorithm with per-utterance updates.

For the simplified system, the performance improvement from the proposed SSNN method is comparable to the case when i-vectors are used. The combination of i-vectors and SSNN leads to further improvements.

On the *very challenging* complete system, the i-vectors were more beneficial, than the SSNN method, which did not bring clear improvement compared to the baseline, but the results from long sentences in figure 2 are encouraging.

The benefit of the proposed technique is that it does not require multi-pass decoding and it relies only on one single test utterance, giving it the potential to become more practical than the i-vector method which needs several utterances for each speaker and the complicated i-vector extraction framework.

Finally, our belief is that the use of Sequence Summarizing Neural Network is not limited to speaker adaptation of DNN. We see it as a generic framework which can produce "summary vectors" for sequential data in general. The SSNN framework is conceptually simpler than recurrent networks, which also support the transfer of information across frames.

The information transfer in recurrent networks is local and the order of frames is important. The signal from more distant frames is hard to pass because it gets changed by each cycle of recurrency. This limitation is less severe in LSTMs, where the memory cells keep inner state across frames. Contrarily, the frame order in averaging is unimportant and all the frames are equally important.

In future we can study the interactions of RNNs and the sentence averaging. Another promising direction is to extend the averaging into history or try to improve the "summary vectors" of short sentences.

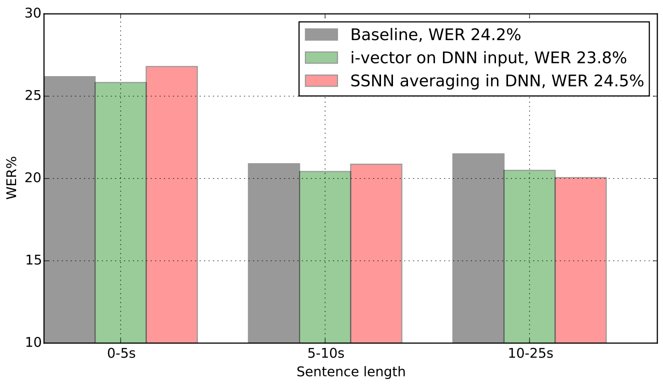

**Fig. 2**. *Performance on dev set, subdivided by sentence lengths (complete recipe, sMBR systems from table 2). The i-vector inputs (estimated per-speaker) are helpful for all sentence lengths, while "summary vectors" (extracted per-sentence) are deletrious for short sentences and helpful for sentences longer than 10 seconds.*

## 7. REFERENCES

[1] Frank Seide, Gang Li, Xie Chen, and Dong Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription," in *Proceedings of ASRU*, 2011.

[2] P. Swietojanski and S. Renals, "Learning hidden unit contributions for unsupervised speaker adaptation of neural network acoustic models," in *Proceedings of SLT*, 2014.

[3] Sabato Marco Siniscalchi, Jinyu Li, and Chin-Hui Lee, "Hermitian polynomial for speaker adaptation of connectionist speech recognition systems," *IEEE Transactions on Audio, Speech & Language Processing*, vol. 21, no. 10, 2013.

[4] Najim Dehak, Patrick Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet, "Front-end factor analysis for speaker verification," *IEEE TASLP*, vol. 19, no. 4, pp. 788–798, 2011.

[5] Ondřej Glembek, Lukáš Burget, and Pavel Matějka, "Voice biometry standard," Proposed draft, Speech@FIT, BUT, Brno, Czech Republic, 2015.

[6] Martin Karafiát, Lukáš Burget, Pavel Matějka, Ondřej Glembek, and Jan Černocký, "iVector-based discriminative adaptation for automatic speech recognition," in *Proceedings of ASRU*, 2011.

[7] George Saon, Hagen Soltau, David Nahamoo, and Michael Picheny, "Speaker adaptation of neural network acoustic models using i-vectors," in *Proceedings of ASRU*, 2013.

[8] Mickael Rouvier and Benoît Favre, "Speaker adaptation of DNN-based ASR with i-vectors: does it actually adapt models to speakers?," in *Proceedings of Interspeech*, 2014.

[9] Vijayaditya Peddinti, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur, "Reverberation robust acoustic modeling using i-vectors with time delay neural networks," in *Proceedings of Interspeech*, 2015.

[10] Penny Karanasou, Yongqiang Wang, Mark JF Gales, and Philip C Woodland, "Adaptation of deep neural network acoustic models using factorised i-vectors," in *Proceedings of Interspeech*, 2014.

[11] Chengzhu Yu, Atsunori Ogawa, Marc Delcroix, Takuya Yoshioka, Tomohiro Nakatani, and John HL Hansen, "Robust i-vector extraction for neural network adaptation in noisy environment," in *Proceefings of Interspeech*, 2015.

[12] Sriram Ganapathy, Samuel Thomas, Dimitrios Dimitriadis, and Steven Rennie, "Investigating factor analysis features for deep neural networks in noisy speech recognition," in *Proceedings of Interspeech*, 2015.

[13] Yajie Miao, Lu Jiang, Hao Zhang, and Florian Metze, "Improvements to speaker adaptive training of deep neural networks," in *Proceedings of SLT*, 2014.

[14] Ehsan Variani, Xin Lei, Erik McDermott, Ignacio Lopez Moreno, and Javier Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *Proceedings of ICASSP*, 2014.

[15] Xiangang Li and Xihong Wu, "Modeling speaker variability using long short-term memory networks for speech recognition," in *Proceedings of Interspeech*, 2015.

[16] Karel Veselý, Arnab Ghoshal, Lukáš Burget, and Daniel Povey, "Sequence-discriminative training of deep neural networks," in *Proceedings of Interspeech*, 2013.

[17] Xavier Glorot and Yoshua Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of AISTATS*, 2010.

[18] Pegah Ghahremani, Bagher BabaAli, Daniel Povey, Korbinian Riedhammer, Jan Trmal, and Sanjeev Khudanpur, "A pitch extraction algorithm tuned for automatic speech recognition," in *Proceedings of ICASSP*, 2014.

[19] G E Hinton, S Osindero, and Y Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, pp. 1527–1554, 2006.

[20] Karel Veselý, Mirko Hannemann, and Lukáš Burget, "Semi-supervised training of deep neural networks," in *Proceedings of ASRU*, 2013.