INTERSPEECH 2017
August 20–24, 2017, Stockholm, Sweden

Tied Hidden Factors in Neural Networks for End-to-End Speaker Recognition

Antonio Miguel, Jorge Llombart, Alfonso Ortega, Eduardo Lleida

ViVoLAB, Aragon Institute for Engineering Research (I3A), University of Zaragoza, Spain
{amiguel,jllombg,ortega,lleida}@unizar.es

Abstract

In this paper we propose a method to model speaker and session variability and able to generate likelihood ratios using neural networks in an end-to-end phrase dependent speaker verification system. As in Joint Factor Analysis, the model uses tied hidden variables to model speaker and session variability and a MAP adaptation of some of the parameters of the model. In the training procedure our method jointly estimates the network parameters and the values of the speaker and channel hidden variables. This is done in a two-step backpropagation algorithm, first the network weights and factor loading matrices are updated and then the hidden variables, whose gradients are calculated by aggregating the corresponding speaker or session frames, since these hidden variables are tied. The last layer of the network is defined as a linear regression probabilistic model whose inputs are the previous layer outputs. This choice has the advantage that it produces likelihoods and additionally it can be adapted during the enrolment using MAP without the need of a gradient optimization. The decisions are made based on the ratio of the output likelihoods of two neural network models, speaker adapted and universal background model. The method was evaluated on the RSR2015 database.

Index Terms: Neural Networks, Joint Factor Analysis, Tied Factor Analysis, Speaker variability, Session variability, Linear Regression Models.

1. Introduction

Deep neural networks (DNNs) have been successfully applied in many speech and speaker recognition tasks in recent years, providing outstanding performances. In speech technologies most of the DNN solutions have used them as classifiers or feature extractors, but in this work we propose to apply them in an end-to-end detection task, where the output of the system is a likelihood score ratio, which after applying a threshold provides good performance without the need of further calibration. Unfortunately, the high number of parameters of DNNs and their tendency to overfit data make that type of detection task difficult to approach. In this work we try to take advantage of the high flexibility of these models and their capacity to learn nonlinear patterns from the input signals. This has required to provide solutions to decrease the overfitting problems and their lack of a measure of uncertainty by adding external control variables to model session and speaker variability, and also proposing Bayesian adaptation and evaluation mechanisms.

Recent approaches to speaker recognition use Joint Factor Analysis (JFA) [1, 2, 3, 4, 5] with GMMs or HMMs as base distributions, i-vector systems [6, 7, 8], neural networks as bottleneck feature extractors for JFA or i-vector systems [9, 10], or neural networks as classifiers to produce posterior probabilities for JFA or i-vector extractors [11, 10, 12]. There have been other approaches to create speaker verification using neural networks by means of LSTM networks [13] to provide sequence to vector compression or to extract total variability latent factors (similar to i-vector) directly from a neural network in [14], in that case a PLDA backend was used in a text independent speaker recognition task. The proposed method has several parallelisms to JFA models since we also encode speaker and session information using latent variables, and the model probability distribution can also be adapted to the speaker data using MAP or Bayesian techniques, but in this paper the model is built on top of an autoencoder neural network as an alternative to other models like GMMs or HMMs. The proposed method is an end-to-end solution since the neural network performs all the processing steps and it provides the likelihood ratio. The autoencoder [15] is a generative model that is trained to predict its input with the restriction that the network has a bottleneck layer of smaller dimension. Its training is unsupervised in terms of frame level phoneme labels, what makes it a candidate to substitute GMMs as the underlying model of the system. In addition, it can be robustly trained using Bayesian methods and its output layer can be probabilistic, as it has been shown in recent works [15, 16, 17]. As we show later in the paper, we can build a system using these probabilistic autoencoders, but performance can be improved by using tied hidden variables to model speaker and session variability.

Speaker factors have been used as a source of additional information for neural networks in speech recognition to improve the performance of speaker independent recognizers [18, 19]. In many cases the latent variables were obtained by using an external system like a GMM based JFA or i-vector extractor, and the network is then retrained to capture this extra information [20]. There have been works were speaker factors or speaker codes are used to enhance speech recognition systems and they were optimized by gradient optimization [20, 21] or initialized using Singular Value Decomposition [22]. We propose a joint factor approach for neural networks to model speaker and session variability, showing that effective improvements can be obtained by using the latent factors with respect to a reference network. A modified two-step backpropagation algorithm is proposed to train the model parameters and the latent factors, first the network weights and factor loading matrices are updated given the current value of the latent variables and then the latent variables are updated. To calculate the gradients of the cost function with respect to the network weights the minibatch samples can be randomly permuted to facilitate convergence, but the gradients with respect to the hidden factors are calculated by aggregating all the corresponding speaker or session frames, since these hidden variables are tied.

This paper is organized as follows. In Section 2 the tied factor analysis model for neural networks is presented. Section 3 discusses the use of autoencoder neural networks for speaker verification. Section 4 presents an experimental study to show the accuracy of the models in phrase dependent speaker recognition. Conclusions are presented in Section 5.

http://dx.doi.org/10.21437/Interspeech.2017-1314

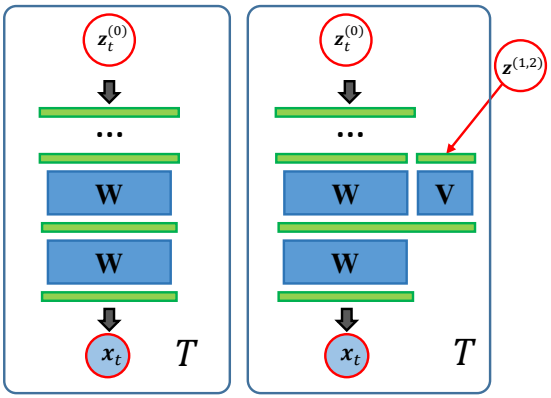

Figure 1: *Decoder in an autoencoder for DNNs and TF-DNN*

2. Tied Hidden Factors in Neural Networks

The concept of tied hidden factors to model contextual or sequence information has appeared in many different contexts like face recognition [23], speech recognition [24, 25], language recognition [26, 27], speaker diarization [28] or audio segmentation [29], but their most prominent field has been speaker recognition with JFA or i-vector approaches [30, 2, 3, 4, 5, 6] The use of these type of global variables has been defined in more general approaches as hierarchical models [31] and more recently in the context of DNNs in [32]. We propose to use two types of tied hidden factors in neural networks to extend previous works with a general algorithm to estimate them. We refer to this model in general as Tied Factor Deep Neural Network (TF-DNN), and for the special case of two factors speaker and session, TF2-DNN. In Figure 1 it is depicted the conceptual difference of the decoder part of a standard DNN autoencoder and a TF-DNN, where $\mathbf{z}_t^{(0)}$ is the bottleneck layer which changes for every frame t, but there are tied variables that affect the output of many samples $\mathbf{x}_t$ of the same session (or file in databases like RSR2015) $\mathbf{z}_f^{(1)}$ and same speaker $\mathbf{z}_s^{(2)}$, displaying a similar the idea to [3, 4] for GMMs.

To define the model, first we need to describe the observed data, $\mathbf{X} = \{\mathbf{x}_t\}_{t=1}^T$ as a sequence of feature vectors $\mathbf{x}_t \in \mathbb{R}^D$ with D the feature dimension. In the proposed TF2-DNN approach we assume that a set of hidden variables encode speaker and session information. The session and speaker latent factors are denoted as $\mathbf{Z}^{(1)} = \{\mathbf{z}_f^{(1)}\}_{f=1}^F$ and $\mathbf{Z}^{(2)} = \{\mathbf{z}_s^{(2)}\}_{s=1}^S$ with $\mathbf{z}_f^{(1)} \in \mathbb{R}^{R^{(1)}}$ and $\mathbf{z}_s^{(2)} \in \mathbb{R}^{R^{(2)}}$, where F and S are the number of sessions and speakers and $R^{(1)}$ and $R^{(2)}$ are the dimension of their respective subspaces. The complete set of hidden variables is denoted as $\mathcal{Z} = (\mathbf{Z}^{(1)}, \mathbf{Z}^{(2)})$, they encode speaker and session information, and since they are unknown we need to estimate them using a labeled dataset. The training data are typically organized by sessions. If sessions are labeled by speaker, then it is straightforward to obtain frame level session labels $\phi_t^{(1)} \in \{1, \dots, F\}$, and speaker labels as $\phi_t^{(2)} \in \{1, \dots, S\}$, so that the training dataset is defined as $\mathcal{D} = \{\mathbf{x}_t, \phi_t^{(1)}, \phi_t^{(2)}\}_{t=1}^T$, for each data sample we need its speaker and session label.

The TF2-DNN is built on top of a regular neural network whose parameters are the weights $\mathbf{W}_l$ and biases $\mathbf{b}_l$ of all the layers $l = 1, \dots, L$. We denote them by $\bar{\mathbf{W}} = \{\mathbf{W}_1, \mathbf{b}_1, \dots, \mathbf{W}_L, \mathbf{b}_L\}$ for a NN with L layers. The layers of the network which are connected to the latent variables are called TF2 layers and have additional parameters and a different output than a standard network. In Figure 1, we show a simplified model of a DNN with standard layers and TF-DNN with a TF2 layer. In the case of linear embedding, a factor loading matrix $\mathbf{V}_l$ is required for each factor and the layer l output is defined as

$$\mathbf{x}_{t,l} = \sigma(\mathbf{W}_l \mathbf{x}_{t,l-1} + \mathbf{b}_l + \mathbf{V}_l^{(1)} \mathbf{z}_f^{(1)} + \mathbf{V}_l^{(2)} \mathbf{z}_s^{(2)}), \qquad (1)$$

where $\mathbf{x}_{t,l-1}$ and $\mathbf{x}_{t,l}$ are the previous layer output and the current layer output, the function $\sigma()$ is the layer nonlinearity, $\mathbf{z}_f^{(1)}$ is the session factor corresponding to the frame $\mathbf{x}_t$, and f is the corresponding file label $f = \phi_t^{(1)}$, and $\mathbf{z}_s^{(2)}$ is the corresponding speaker factor and s is the speaker label $s = \phi_t^{(2)}$. The set of all the network parameters is denoted as $\mathbf{\Theta} = (\bar{\mathbf{W}}, \bar{\mathbf{V}})$.

Given a cost function $J(\mathcal{D}, \mathcal{Z}, \mathbf{\Theta})$ that can be evaluated for some training data, $\mathcal{D}$, and an instance of the unknown parameters $\mathcal{Z}$ and $\mathbf{\Theta}$, we can define an optimization method to minimize the cost. In this work we have used gradient based optimization since it can be scaled to larger datasets and still be tractable. To estimate both the network weights and the tied latent factors Algorithm 1 is proposed. This algorithm optimizes both sets of variables in an alternate way like in coordinate descent type algorithms. In [4] the alternate E and M steps had the same motivation, since the E step considered the likelihood as the objective and the latent variables were optimized in a search process given the other parameters fixed. The gradients with respect to the network parameters $\mathbf{\Theta}$ in the step 2.1 are computed as usual gradients since the hidden variables are given as argument with their current value and they can be interpreted as external information to the network. The gradients with respect to the tied hidden variables have to be considered more carefully, since they have to be calculated by aggregating all the corresponding speaker or session frames since they are tied.

Algorithm 1 Training algorithm for two tied hidden factor neural network (TF2-DNN)

Input: $\mathcal{D}$: Acoustic features $\mathbf{X}$, frame level session labels and frame level speaker labels $\phi^{(1)}, \phi^{(2)}$, and prior values for the initialization λ, learning rate values for the unknown parameters α, and number of epochs N

Output: Estimations for hidden factors $\mathcal{Z} = (\mathbf{Z}^{(1)}, \mathbf{Z}^{(2)})$ and the network parameters $\mathbf{\Theta} = (\bar{\mathbf{W}}, \bar{\mathbf{V}})$: the neural network weighs and factor loading matrices

1. Initialization

Initialize all the unknown weights and latent factors randomly following their prior distribution:

$\mathbf{\Theta} \sim \mathcal{N}(\mathbf{0}, \lambda_1 \mathbf{I}),\ \mathbf{z}^{(1)} \sim \mathcal{N}(\mathbf{0}, \lambda_2 \mathbf{I}),\ \mathbf{z}^{(2)} \sim \mathcal{N}(\mathbf{0}, \lambda_3 \mathbf{I})$

2. Two step backpropagation

for $n \leftarrow 1$ **to** N **do**

2.1 Step 1, backpropagation parameters: $\mathbf{\Theta}$:

Minibatch b updates: frames t_b are selected randomly:

$\mathcal{D}_b \leftarrow \{\mathbf{x}_t, \phi_t^{(1)}, \phi_t^{(2)} | t \in t_b\}$,

$\mathcal{Z}_b \leftarrow \{\mathbf{z}_f^{(1)}, \mathbf{z}_s^{(2)} | t \in t_b, f = \phi_t^{(1)}, s = \phi_t^{(2)}\}$

$\mathbf{\Theta} \leftarrow \mathbf{\Theta} - \alpha_1 \nabla_{\mathbf{\Theta}} J(\mathcal{D}_b, \mathcal{Z}_b, \mathbf{\Theta})$

2.2 Step 2, backpropagation hidden variables $\mathbf{z}^{(1)}$, $\mathbf{z}^{(2)}$

Using expressions (2), (3) for all speaker s and sessions f

$\mathbf{z}_f^{(1)} \leftarrow \mathbf{z}_f^{(1)} - \alpha_2 \nabla_{\mathbf{z}_f^{(1)}} J(\mathcal{D}, \mathcal{Z}, \mathbf{\Theta})$

$\mathbf{z}_s^{(2)} \leftarrow \mathbf{z}_s^{(2)} - \alpha_3 \nabla_{\mathbf{z}_s^{(2)}} J(\mathcal{D}, \mathcal{Z}, \mathbf{\Theta})$

end

Then the gradients for session f and speaker s factors are

$$\nabla_{\mathbf{z}_f^{(1)}} J(\mathcal{D}, \mathcal{Z}, \mathbf{\Theta}) = \sum_{t|\phi_t^{(1)}=f} \nabla_{\mathbf{z}^{(1)}} J(\mathbf{x}_t, \mathbf{z}^{(1)}_{\phi_t^{(1)}}, \mathbf{z}^{(2)}_{\phi_t^{(2)}}, \mathbf{\Theta}) \quad (2)$$

$$\nabla_{\mathbf{z}_s^{(2)}} J(\mathcal{D}, \mathcal{Z}, \mathbf{\Theta}) = \sum_{t|\phi_t^{(2)}=s} \nabla_{\mathbf{z}^{(2)}} J(\mathbf{x}_t, \mathbf{z}^{(1)}_{\phi_t^{(1)}}, \mathbf{z}^{(2)}_{\phi_t^{(2)}}, \mathbf{\Theta}). \quad (3)$$

3. Neural network end-to-end speaker verification system

The autoencoder [15, 16] is a generative model that is trained to predict its input with the restriction that the network has a bottleneck layer of smaller dimension. Then, the system has two parts: the first part, encoder, learns how to compress the information and the second part, decoder, learns how to reconstruct the signal. To adapt this type of network to the task of speaker recognition we need to compute likelihoods of the observed data $\mathbf{x}_t$. In [15] the bottleneck layer of the autoencoder was considered analogous to the hidden variable $\mathbf{z}_t$ in a factor analysis model [33]. Then, the encoder part was associated to a variational approximation to the posterior distribution $q(\mathbf{z}_t|\mathbf{x}_t)$, and the decoder part could be associated to the likelihood of the observed data given the hidden variable by parametrizing a probabilistic model using the network outputs $p(\mathbf{x}_t|\mathbf{z}_t)$. To follow with the previous section notation for the latent factors, [4], the bottleneck layer of the autoencoder is denoted as $\mathbf{z}_t^{(0)}$, since it encodes intra-frame information [34, 33] and it is the lowest level in the hierarchy: frame (0), session (1), speaker (2). We can see that other levels could be added easily.

3.1. Linear Regression probability model

In this paper we use the same parametrization mechanism as in [15, 17] to define that the last layer provides the mean vector of a Gaussian distribution, which we combine with the following idea: if the last layer is a linear function, $\mathbf{x}_{t,L} = \mathbf{W}_L\mathbf{x}_{t,L-1} + \mathbf{b}_L$, the likelihood can also be interpreted as a linear regression probability model whose regression coefficient matrix $\mathbf{B}$ is the last layer weight matrix $\mathbf{W}_L$ as

$$p(\mathbf{x}_t|\mathbf{z}_t^{(0)}) = \mathcal{N}(\mathbf{x}_{t,L}, \mathbf{\Psi}) = \mathcal{N}(\mathbf{W}_L\mathbf{x}_{t,L-1} + \mathbf{b}_L, \mathbf{\Psi}), \quad (4)$$

where the output $\mathbf{x}_{t,L}$ acts as mean and we can define an arbitrary covariance matrix $\mathbf{\Psi}$.

The following steps could be carried out for the bias parameter $\mathbf{b}_L$ and the covariance matrix $\mathbf{\Psi}$ as well, but to keep the notation simpler [35], and focus on the most important parameters, the weigths $\mathbf{B} = \mathbf{W}_L$, we derive the network adaptation mechanism for this simpler distribution

$$p(\mathbf{x}_t|\mathbf{y}_t) = \mathcal{N}(\mathbf{B}\mathbf{y}_t, \beta^{-1}\mathbf{I}), \quad (5)$$

where $\beta^{-1}\mathbf{I}$ is the covariance matrix, now controlled by a single parameter and we denote the outputs of the previous layer as $\mathbf{y}_t$ for simplicity, with $\mathbf{y}_t = \mathbf{x}_{t,L-1} = f(\mathbf{z}_t^{(0)})$ using the decoder part of the network except the last layer.

The analogy in expressions (4) and (5) allows to estimate the value of $\mathbf{W}_L$ using a probabilistic approach if we let the rest of the network parameters and hidden variables unchanged, what makes easy to apply ML, MAP or Bayesian estimation techniques. Given some training data $\mathbf{X}$ organized by rows, and the output previous to the last layer $\mathbf{Y}$, the ML estimator is equivalent to minimize square error, MSE, and is obtained as

$$\mathbf{B}^{ML} = (\mathbf{Y}^\intercal\mathbf{Y})^{-1}\mathbf{Y}^\intercal\mathbf{X}. \quad (6)$$

The ML estimator can have problems when inverting the matrix if it is ill-conditioned. To solve that we can impose a penalty to the weights $\mathbf{B}$ by assuming a Gaussian prior for them $\mathbf{B} \sim \mathcal{N}(\mathbf{B}_0, \lambda_0^{-1}\mathbf{I})$, which makes the optimization of the posterior distribution $p(\mathbf{B}|\mathbf{X})$ equivalent to an L2 regularization [17], this is the MAP estimator

$$\mathbf{B}^{MAP} = (\beta\mathbf{Y}^\intercal\mathbf{Y} + \lambda_0\mathbf{I})^{-1}(\beta\mathbf{Y}^\intercal\mathbf{X} + \lambda_0\mathbf{B}_0), \quad (7)$$

which in the case the prior mean is zero it is usually expressed as

$$\mathbf{B}^{MAP} = (\mathbf{Y}^\intercal\mathbf{Y} + \frac{\lambda_0}{\beta}\mathbf{I})^{-1}\mathbf{Y}^\intercal\mathbf{X}. \quad (8)$$

The fully Bayesian approach [35] provides a posterior distribution for the weights given the priors and the observed data that follows a normal distribution, whose mean has the same value as the MAP estimation in (7)

$$p(\mathbf{B}|\mathbf{Y}) = \mathcal{N}(\mathbf{B}_N, \Sigma_N) \quad (9)$$

$$\mathbf{B}_N = \Sigma_N(\beta\mathbf{Y}^\intercal\mathbf{X} + \lambda_0\mathbf{B}_0) \quad (10)$$

$$\Sigma_N^{-1} = (\beta\mathbf{Y}^\intercal\mathbf{Y} + \lambda_0\mathbf{I}). \quad (11)$$

3.2. UBM training, speaker enrolment and trial evaluation

Once the building blocks of the model have been established, we describe all the basic steps involved in a speaker recognition system: UBM training, speaker enrolment, and trial evaluation.

The universal background model (UBM) is trained using Algorithm 1 after a random initialization of all the unknown parameters and latent factors. The labels to assign training frames to the speaker and latent factors must be supplied to the algorithm. When the UBM is trained we extract the sums

$$S_{yy} = \mathbf{Y}^\intercal\mathbf{Y} = \sum_t \mathbf{y}_t\mathbf{y}_t^\intercal, \quad S_{yx} = \mathbf{Y}^\intercal\mathbf{X} = \sum_t \mathbf{y}_t\mathbf{x}_t^\intercal, \quad (12)$$

which are the sufficient statistics needed to make adaptations at enrolment time without the need of processing the whole database each time. $\mathbf{y}_t$ and $\mathbf{x}_t$ are column vectors corresponding to frame t. Then, $\mathbf{B}^{ubm}$ can be obtained from the stats.

To enrol a speaker in the system in the context of the proposed method requires to create an adapted network to the samples used for enrolment, which are a small number compared to the UBM. Two mechanism are available in this model for this. The first one is to adapt the speaker latent factor by using step 2.2 of the algorithm for a number of iterations using the enrolment data and using as initial values the UBM parameters. In this case step 2.1 would not be applied since the network weights have to remain fixed. The second mechanism is more similar to MAP in JFA systems, two possible adaptations are proposed given the previous linear regression expressions. One option is to consider the prior mean as the UBM value $\mathbf{B}_0 = \mathbf{B}^{ubm}$, then using expression (10) the maximum of the posterior distribution $p(\mathbf{B}^{spk}|\mathbf{Y}_{spk}, \mathbf{B}_0 = \mathbf{B}^{ubm})$

$$\mathbf{B}^{spk} = (\beta S_{yy}^{spk} + \lambda_0\mathbf{I})^{-1}(\beta S_{yx}^{spk} + \lambda_0\mathbf{B}^{ubm}). \quad (13)$$

Other option is to consider the posterior distribution given both the enrolment and the UBM data $p(\mathbf{B}^{spk}|\mathbf{Y}_{ubm}, \mathbf{Y}_{spk})$, with the prior mean equal to zero. To control the weight of the UBM samples with respect to the enrolment we introduce an interpolation factor α

$$\mathbf{B}^{spk} = \quad (14)$$
$$(\alpha S_{yy}^{ubm} + (1-\alpha)S_{yy}^{spk} + \frac{\lambda_0}{\beta}\mathbf{I})^{-1}(\alpha S_{yx}^{ubm} + (1-\alpha)S_{yx}^{spk}).$$

Finally for trial evaluation we evaluate the likelihood ratio

$$\Lambda = \frac{p(\mathbf{X}|\mathbf{Y}_{ubm}, \mathbf{Y}_{spk})}{p(\mathbf{X}|\mathbf{Y}_{ubm})}, \quad (15)$$

where likelihoods are calculated using expression (4), $\mathbf{B}^{ubm}$ in the denominator is estimated using (10) and the sufficient stats, (12), for all the UBM data $\mathbf{Y}_{ubm}$. $\mathbf{B}^{spk}$ in the numerator is estimated using (14) and the speaker and UBM data, $\mathbf{Y}_{spk}, \mathbf{Y}_{ubm}$, (since it performed better than (13) in preliminary experiments).

3.3. Bayesian inference

Recent advances on applying Bayesian estimation techniques to DNNs have been shown to be effective against overfitting and to deal with uncertainty [36, 15, 17]. To avoid overfitting when data size is small, we propose to use dropout layers, [37], interleaved with the TFA2 layers, as an alternative to the variational approach [16]. We train the network using the proposed algorithm with the dropout Bernoulli distribution, $\xi \sim Be(p)$, which switches off some of the layer outputs with probability p. Then we perform the trial evaluation by sampling

$$p(\mathbf{x}_t|\mathbf{z}_t^{(0)}) = \int p(\mathbf{x}_t|\mathbf{z}_t^{(0)}, \xi) p(\xi) d\xi \simeq \frac{1}{L} \sum_{\xi_l \sim Be(p)} p(\mathbf{x}_t|\mathbf{z}_t^{(0)}, \xi_l). \quad (16)$$

4. Experiments

The experiments have been conducted on the RSR2015 part I text dependent speaker recognition database [38]. The speakers are distributed in three subsets: bkg, dev and eval. We have only used background data (bkg) to train the UBMs, which can be phrase independent or phrase dependent, as in [8, 3, 4]. The evaluation part is used for enrolment and trial evaluation. The dev part was not used in these experiments and files were not rejected because of low quality. Speaker models are build using 3 utterances for each combination of speaker and phrase (1708 for males and 1470 for females). For testing we have selected the trials using the same phrase as the model, called impostor-correct in [38]: 10244(tgt) + 573664(non) = 583908 male trials; 8810(tgt) + 422880(non) = 431690 female trials. We use the database 16kHz signals to extract 20 MFCCs and their first and second derivatives. Then, an energy based voice activity detector is used and data are normalized using short term Gaussianization. In the experiments in this work the speaker factor is a speaker-phrase combination as in [3]. To train the autoencoders in this work we used the same DNN architecture in all of the experiments of 4 hidden layers of 500 units, softplus nonlinearities [17] and a bottleneck layer of 15 units, which makes the dimension $R^{(0)}$ four times smaller than the feature dimension of 60, and finally the linear regression layer. The weight and factor loading matrices were updated using Adam [39] and the cost function was the MSE. The likelihood (4) in the experiments was calculated using bias and diagonal covariance matrix.

A set of experiments was performed using Theano [40] to evaluate the model using gender dependent and phrase independent UBMs. We compared the DNN which updates the last layer using (14) to the TF2-DNN which also updates the last layer and in addition includes speaker and session factors. The results in Table 1 show both DNN systems performing under 1% EER for the male and female tasks, but the performance is greater in the case of models using tied factors in the model. In the paper we exposed some parallelism between the DNN and a GMM both adapted with MAP. We can see in the experiments that the range of EER achieved is also comparable to GMMs in other works [10]. And the relative improvement provided by the TF2-DNN with respect to the DNN system is also similar to [4], although in that case the files were processed at 8kHz.

Table 1: *Experimental results on RSR2015 part I [38] impostor-correct, showing EER% and NIST 2008 and 2010 min costs.*

			Male		
System	$R^{(1)}$	$R^{(2)}$	EER%	det08	det10
DNN	-	-	0.65	0.037	0.155
TF2-DNN	15	50	0.25	0.016	0.086
		75	0.31	0.017	0.080
		100	0.30	0.017	0.085
	25	50	0.29	0.016	0.075
		75	**0.25**	0.015	0.075
		100	0.31	0.017	0.069
			Female		
	$R^{(1)}$	$R^{(2)}$	EER%	det08	det10
DNN	-	-	0.50	0.021	0.084
TF2-DNN	15	50	0.17	0.006	0.019
		75	0.16	0.006	0.026
		100	0.17	0.006	0.030
	25	50	0.15	0.007	0.028
		75	**0.13**	0.006	0.028
		100	0.19	0.007	0.021

In RSR2015, phrase dependent UBMs can be more specific, but there are less data available for the UBM, which makes difficult to train a DNN with many layers. For that scenario we propose the use of dropout [37] and to approximate the likelihoods using (16). We performed some experiments using the female subset and phrase dependent UBMs. A dropout layer was interleaved in the encoder and the decoder, with $p = 0.05$. The TF2-DNN had as dimensions $R^{(1)} = 5$ and $R^{(2)} = 20$) and the system provided a 0.11% of EER. This preliminary experiment showed us that dropout and other Bayesian techniques can mitigate part of the effect of overfitting of large DNNs when learning small datasets, and the system still can provide well calibrated scores in the context of these models.

5. Conclusions

In this paper we present an end-to-end method for speaker recognition based on neural networks, using tied hidden variables to model speaker and session variability and a MAP and Bayesian techniques to enrol and evaluate trials. The last layer of the network is defined as a linear regression probabilistic model that can be adapted during the enrolment so that the model can calculate likelihood ratios to decide the trial evaluations. To estimate the model parameters and the hidden variables a two-step backpropagation algorithm is used. We have tested the models in the text dependent speaker recognition database RSR2015 part I providing competitive results with respect to previous approaches.

6. Acknowledgements

This work is supported by the Spanish Government and European Union (project TIN2014–54288–C4–2–R), and by the European Commission FP7 IAPP Marie Curie Action GA–610986. We gratefully acknowledge the support of NVIDIA Corporation with the donation of a Titan X GPU used for this research.

7. References

[1] S.-C. Yin, R. Rose, and P. Kenny, "A Joint Factor Analysis Approach to Progressive Model Adaptation in Text-Independent Speaker Verification," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 15, no. 7, pp. 1999–2010, Sep. 2007.

[2] P. Kenny, P. Ouellet, N. Dehak, V. Gupta, and P. Dumouchel, "A Study of Interspeaker Variability in Speaker Verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 5, pp. 980–988, Jul. 2008.

[3] P. Kenny, T. Stafylakis, J. Alam, P. Ouellet, and M. Kockmann, "Joint factor analysis for text-dependent speaker verification," in *Proc. Odyssey Workshop*, 2014, pp. 1–8.

[4] A. Miguel, A. Ortega, E. Lleida, and C. Vaquero, "Factor analysis with sampling methods for text dependent speaker recognition." Proc. Interspeech.

[5] T. Stafylakis, P. Kenny, M. J. Alam, and M. Kockmann, "Speaker and Channel Factors in Text-Dependent Speaker Recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 1, pp. 65–78, 1 2016.

[6] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-End Factor Analysis For Speaker Verification," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 19, no. 14, pp. 788–798, May 2010.

[7] J. Villalba and N. Brümmer, "Towards Fully Bayesian Speaker Recognition: Integrating Out the Between-Speaker Covariance," in *Interspeech 2011*, Florence, 2011, pp. 28–31.

[8] T. Stafylakis, P. Kenny, P. Ouellet, J. Perez, M. Kockmann, and P. Dumouchel, "Text-dependent speaker recogntion using plda with uncertainty propagation," in *Proc. Interspeech*, Lyon, France, August 2013.

[9] Y. Liu, Y. Qian, N. Chen, T. Fu, Y. Zhang, and K. Yu, "Deep feature for text-dependent speaker verification," *Speech Communication*, vol. 73, no. C, pp. 1–13, 10 2015.

[10] H. Zeinali, L. Burget, H. Sameti, O. Glembek, and O. Plchot, "Deep neural networks and hidden markov models in i-vector-based text-dependent speaker verification," in *Odyssey-The Speaker and Language Recognition Workshop*, 2016.

[11] S. Dey, S. Madikeri, M. Ferras, and P. Motlicek, "Deep neural network based posteriors for text-dependent speaker verification," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 3 2016, pp. 5050–5054.

[12] H. Zeinali, H. Sameti, L. Burget, J. Cernocky, N. Maghsoodi, and P. Matejka, "i-vector/hmm based text-dependent speaker verification system for reddots challenge," in *Proc. Interspeech*. ISCA, 2016.

[13] G. Heigold, I. Moreno, S. Bengio, and N. M. Shazeer, "End-to-end text-dependent speaker verification," in *Proc. ICASSP*, 2016.

[14] S. Garimella and H. Hermansky, "Factor analysis of auto-associative neural networks with application in speaker verification," *IEEE transactions on neural networks and learning systems*, vol. 24, no. 4, pp. 522–528, 2013.

[15] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proc. ICLR*, no. 2014, 2013.

[16] D. P. Kingma, T. Salimans, and M. Welling, "Variational dropout and the local reparameterization trick," in *Proc. NIPS*, 2015.

[17] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural network," in *Proc. ICML*, 2015, pp. 1613–1622.

[18] A. Senior and I. Lopez-Moreno, "Improving dnn speaker independence with i-vector inputs," in *Proc. ICASSP*, 2014.

[19] S. Xue, O. Abdel-Hamid, H. Jiang, L. Dai, and Q. Liu, "Fast adaptation of deep neural network based on discriminant codes for speech recognition," *IEEE/ACM Transactions on Audio, Speech and Language Processing*, vol. 22, no. 12, pp. 1713–1725, 2014.

[20] L. Samarakoon and K. C. Sim, "Factorized hidden layer adaptation for deep neural network based acoustic modeling," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 12, pp. 2241–2250, 2016.

[21] Z. Huang, J. Tang, S. Xue, and L. Dai, "Speaker adaptation of rnn-blstm for speech recognition based on speaker code," in *Proc. ICASSP*. IEEE, 2016, pp. 5305–5309.

[22] S. Xue, H. Jiang, L. Dai, and Q. Liu, "Unsupervised speaker adaptation of deep neural network based on the combination of speaker codes and singular value decomposition for speech recognition," in *Proc. ICASSP*. IEEE, 2015, pp. 4555–4559.

[23] S. Prince, J. Warrell, J. Elder, and F. Felisberti, "Tied Factor Analysis for Face Recognition across Large Pose Differences," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 6, pp. 970–984, 6 2008.

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

[25] P. Kenny, G. Boulianne, and P. Dumouchel, "Eigenvoice modeling with sparse training data," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 3, pp. 345–354, May 2005.

[26] D. Martínez, O. Plchot, L. Burget, G. Ondrej, and P. Matejka, "Language Recognition in iVectors Space," in *Proc. Interspeech*, Florence, Italy, 2011.

[27] D. Martínez, E. Lleida, A. Ortega, and A. Miguel, "Prosodic Features and Formant Modeling for an iVector-Based Language Recognition System," in *ICASSP*, Vancouver, Canada, 2013.

[28] C. Vaquero, A. Ortega, A. Miguel, and E. Lleida, "Quality Assessment for Speaker Diarization and Its Application in Speaker Characterization," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 4, pp. 816–827, Apr. 2013.

[29] D. Castan, A. Ortega, J. Villalba, A. Miguel, and E. Lleida, "SEGMENTATION-BY-CLASSIFICATION SYSTEM BASED ON FACTOR ANALYSIS," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2013.

[30] P. Kenny, "Joint Factor Analysis of Speaker and Session Variability : Theory and Algorithms," CRIM, Montreal, CRIM-06/08-13, Tech. Rep., 2005.

[31] A. Gelman and J. Hill, *Data Analysis Using Regression and Multilevel/Hierarchical Models*, ser. Analytical Methods for Social Research. Cambridge University Press, 2006.

[32] D. Tran, R. Ranganath, and D. M. Blei, "Deep and Hierarchical Implicit Models," Feb. 2017. [Online]. Available: http://arxiv.org/abs/1702.08896

[33] Z. Ghahramani and M. J. Beal, "Variational Inference for Bayesian Mixtures of Factor Analysers."

[34] Z. Ghahramani and G. Hinton, "The EM algorithm for mixtures of factor analyzers," Dept. of Comp. Sci., Univ. of Toronto, Toronto, Tech. Rep. 1, 1996. [Online]. Available: http://www.learning.eng.cam.ac.uk/zoubin/papers/tr-96-1.pdf

[35] C. M. Bishop, "A New Framework for Machine Learning," pp. 1–24, 2008.

[36] M. Welling and Y. W. Teh, "Bayesian learning via stochastic gradient langevin dynamics," in *Proc. ICML*, 2011, pp. 681–688.

[37] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[38] A. Larcher, K. A. Lee, B. Ma, and H. Li, "Text-dependent Speaker Verification: Classifiers, databases and RSR2015," *Speech Communication*, vol. 60, pp. 56–77, 2014.

[39] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2014.

[40] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio, "Theano: new features and speed improvements," Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.