END-TO-END DNN BASED SPEAKER RECOGNITION INSPIRED BY I-VECTOR AND PLDA

Johan Rohdin, Anna Silnova, Mireia Diez, Oldřich Plchot, Pavel Matějka, Lukáš Burget

Brno University of Technology, Brno, Czechia

arXiv:1710.02369v1 [eess.AS] 6 Oct 2017

ABSTRACT

Recently several end-to-end speaker verification systems based on deep neural networks (DNNs) have been proposed. These systems have been proven to be competitive for text-dependent tasks as well as for text-independent tasks with short utterances. However, for text-independent tasks with longer utterances, end-to-end systems are still outperformed by standard i-vector + PLDA systems. In this work, we develop an end-to-end speaker verification system that is initialized to mimic an i-vector + PLDA baseline. The system is then further trained in an end-to-end manner but regularized so that it does not deviate too far from the initial system. In this way we mitigate overfitting which normally limits the performance of end-to-end systems. The proposed system outperforms the i-vector + PLDA baseline on both long and short duration utterances.

Index Terms— Speaker verification, DNN, end-to-end

1. INTRODUCTION

In recent years, there have been many attempts to take advantage of Deep Neural networks (DNNs) in speaker verification. This is motivated by large performance improvements brought by DNNs to many other pattern recognition tasks such as speech recognition [1] and face recognition [2].

Most of the attempts aim at replacing or improving one of the components of an i-vector + PLDA system (feature extraction, calculation of sufficient statistics, i-vector extraction or PLDA) with a neural network. For example, DNN bottleneck features instead of conventional MFCC features [3], DNN acoustic models instead of Gaussian mixture models for extraction of sufficient statistics [4], DNNs for either complementing PLDA [5, 6] or replacing it [7]. More ambitiously, so called “d-vectors” [8], replace both the sufficient statistics calculation and i-vector extraction by a single neural network that takes features from the whole utterance as inputs and outputs a vector representation of the utterance.

More recently, there has been several attempts to replace the whole speaker recognition chain with one neural network [9, 10, 11, 12]. Such systems are usually referred to as “end-to-end” systems. This idea is appealing because all parameters of the system are trained jointly for the the intended task. End-to-end systems have been proven competitive for some text-dependent tasks [9] as well as some text-independent tasks with very short test utterances and an abundance of training data [10]. However, on text-independent tasks with longer utterances, end-to-end systems are still being outperformed by standard i-vector + PLDA systems [10].

This project has received funding from the European Unions Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie under grant agreement No. 748097 and the Marie Sklodowska-Curie co-financed by the South Moravian Region under grant agreement No. 665860. This research was also supported by the Google Faculty Research Award program.

In this work, we develop an end-to-end speaker verification system that is initialized to mimic an i-vector + PLDA baseline. The system consists of a neural network (NN) module for extraction of sufficient statistics (**f2s**), an NN module for extraction of i-vectors (**s2i**) and finally, a discriminative PLDA (DPLDA) model [13, 14] for producing scores. These three modules are first developed individually so that they mimic the corresponding part of the i-vector + PLDA baseline. After the modules have been trained individually they are combined and the system is further trained in an end-to-end manner. During the end-to-end training, we regularize the model parameters towards the initial parameters so that they do not deviate too far from them. In this way the system is prevented from becoming too different from the original i-vector + PLDA baseline which reduces the risk of overfitting. Additionally, by first developing the three modules individually, we can more easily find good architectures for them as well as detect difficulties to be aware of in the end-to-end training.

We evaluate the system on three different data sets that are derived from previous NIST SREs. The three test sets contain speech from various languages and were designed to test the performance both on long (longer than two minutes) and short (shorter than 40s) utterances. The achieved results show that the proposed system outperforms both generatively and discriminatively trained i-vector + PLDA baselines.

2. DATASETS AND BASELINE SYSTEMS

2.1. Datasets

We followed the design of the PRISM[15] dataset in the sense of splitting the data into **training** and test sets. The PRISM set contains data from the following sources: NIST SRE 2004 - 2010 (also known as MIXER collections), Fisher English and Switchboard. During training of the end-to-end system initialization, we used the female portion of the NIST SRE'10 telephone condition (condition 5) to independently tune the performance of the blocks A and B in Figure 1.

We report results on three different datasets:

- The female part of the **PRISM language** condition[1] that is based on original (long) telephone recordings from NIST SRE 2005 - 2010. It contains trials from various languages, including the cross-language trials.
- The **short lang** condition (also containing only female trials) is derived from the PRISM language condition by taking multiple short cuts from each original recording. Durations of the speech in the cuts reflect the evaluation plan for NIST SRE'16 - more precisely we based our cuts on the actual detected speech in the SRE'16 labeled development data. We chose the cuts to follow the uniform distribution:
  - Enrollment between 25-50 seconds of speech

[1] For detailed description, please see section B, paragraph 4 of[15].

– Test between 3-40 seconds of speech

We split the resulting set into two equally large disjoint sets where speakers do not overlap. We used one part as our **dev** set for tuning the performance of the DPLDA and the end-to-end system. The other part was used for evaluation only. It should be noted that, for simplicity, we test only on single-enrollment trials unlike in our SRE'16 system description where we include multi-enrollment trials [16].

- Additionally, we report the results on the single-enrollment trials of the NIST SRE'16 evaluation set (both males and females).

2.2. Generative and Discriminative Baselines

As features we used 60-dimensional spectral features (20 MFCCs, including C_0, augmented with their Δ and $\Delta\Delta$ features). The features were short-term mean and variance normalized over a 3 second sliding window.

Both PLDA and DPLDA are based on i-vectors [17] extracted by means of UBM with 2048 diagonal covariance components. Both UBM and i-vector extractor with 600 dimensions are trained on the **training** set. For training our generative (PLDA) and discriminative (DPLDA [13]) baseline systems, we used only telephone data from the **training** set and we also included short cuts derived from portion of our training data that come from non-English or non-native-English speakers. The duration of the speech in cuts follows the uniform distribution between 10-60 seconds. The cuts comprise of 22766 segments out of total 85858. Finally, we augmented the training data with labeled development data from NIST SRE'16.

PLDA: We used the standard PLDA recipe, when i-vectors are mean (mean is calculated using all training data) and length normalized. Then the Linear Discriminant Analysis (LDA) is applied prior PLDA training, decreasing dimensions of i-vectors from 600 to 250. We did not perform any additional domain adaptation or score normalization. We also filtered the training data in such a way that each speaker has at least six utterances which reduces it to the total of 62994 training segments.

Discriminative PLDA: The DPLDA baseline model was trained on the full batch of i-vectors by means of LBFGS optimizing the binary cross-entropy on the training data. We used the **dev** set to tune a single constant that is used for L2 regularization imposed on all parameters except the constant (k in eq. 1).

All i-vectors were mean (mean was calculated using all training data available) and length normalized. After the mean normalization, we performed LDA, decreasing the dimensionality of vectors to 250. As an initialization of DPLDA training, we used a corresponding PLDA model. During the DPLDA training, we set the prior probability of target trials to reflect the SRE'16 evaluation operating point (exactly in the middle between the two operating points of SRE'16 DCF[18]).

3. PROPOSED END-TO-END DNN ARCHITECTURE

In this section we describe the proposed end-to-end architecture. The system is depicted in Figure 1. We devote one subsection to the *features to statistics* module, one subsection for the *statistics to i-vectors* module, one subsection to the *DPLDA* module, and finally one subsection for describing the combination of the three modules. In Section 5, we analyse these components more in detail.

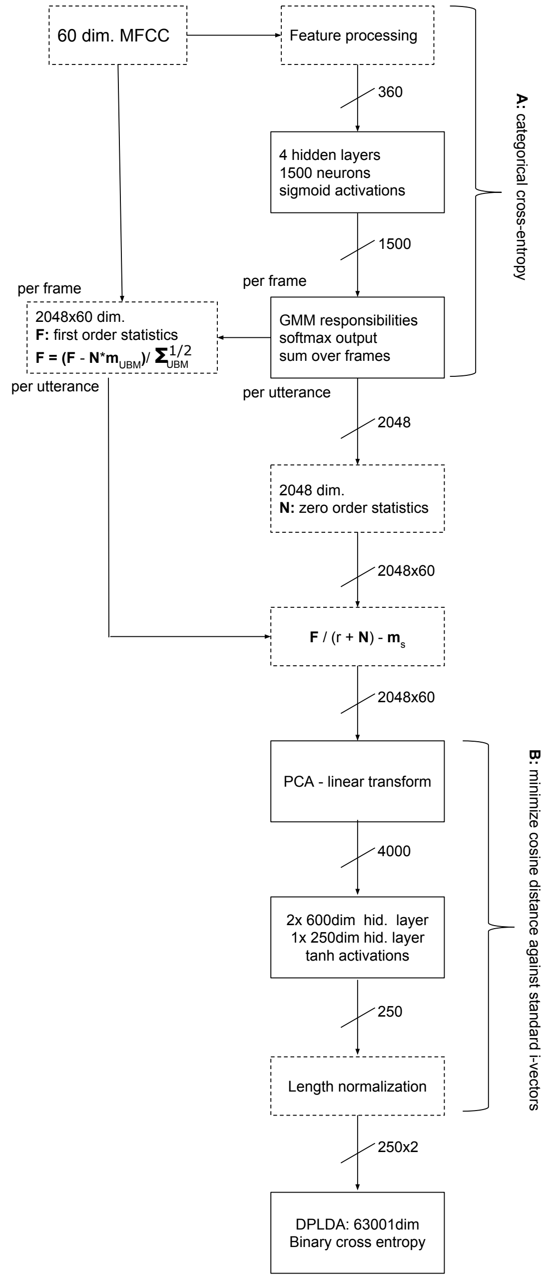

Fig. 1. Block diagram of the End-to-End system. Part **A** corresponds to the UBM that converts features to GMM responsibilities. By adding the next two blocks we obtain first order statistics (**f2s**). Part **B** (**s2i**) simulates the i-vector extraction followed by LDA and length normalization. Parameters in solid line blocks are meant to be trained, while outputs of the dashed blocks are directly computed.

3.1. Features to sufficient statistics

The first module of the end-to-end system translates a sequence of feature vectors into a vector of zeroth and first order statistics. We will denote this module as **s2i**. This module consists, of a network that predicts a vector of GMM responsibilities (posteriors) for each frame of the input utterance (Block A in Figure 1), followed a layer for pooling the frames into sufficient statistics. The network that predicts responsibilities consists of four hidden layers with sigmoid activation functions and a softmax output layer. All hidden layers have 1500 neurons while the output layer has 2048 elements which corresponds to the number of components in our baseline GMM-UBM. We train this network with stochastic gradient descent (SGD) to optimize the categorical cross-entropy with the GMM-UBM posteriors as targets.

As input to the network, the acoustic features described in Section 2.2 are preprocessed as follows. For each frame, a window of 31 frames around the current frame (i.e. +/- 15 frames) is considered. In this window, the temporal trajectory of each feature coefficient is weighted by a Hamming window and projected into first 6 DCT bases (including C_0) [19]. This results in a $6 \times 60 = 360$-dimensional input to the network for each frame.

Once the network predicting responsibilities is trained, we add one more layer that outputs a vector of first order sufficient statistics for the whole utterance. The input to this layer is a matrix of frame-by-frame responsibilities coming from the previous softmax layer and a matrix of original acoustic features without any preprocessing. This layer is not trained but designed in such way that it exactly reproduces the standard calculation of sufficient statistics (first and second order) used in i-vector extraction. It should be noted that expanding the features should in principe not be neceassy in order to predict the GMM-UBM posteriors since these are calculated from original features. However, by using the expanded features, we hope that we can gain further improvements in the end-to-end training.

3.2. Sufficient statistics to i-vectors

The second module of the end-to-end system is trained to mimic the i-vector extraction from the sufficient statistics (Block B in Figure 1). We will denote this module as **s2i**. The input sufficient statistics were first converted into MAP adapted supervectors [20]. To overcome the computational problems that would arise when using the 122880 dimensional supervector as input to the NN, the supervector were reduced by PCA into a 4000 dimensional space. The NN consists of two 600 dimensional hidden layers, with tangens hyperbolicus (tanh) activation functions. The last layer of the NN is designed to produce length normalized 250 dimensional i-vectors. As training objective, we use the average cosine distance between NN outputs and LDA reduced and length-normalized reference i-vectors. The NN is trained with SGD and L1 regularization.

3.3. i-vectors to scores (DPLDA)

The final component of the end-to-end system is a discriminative PLDA (DPLDA) [13, 14] model. The DPLDA model is based on the fact that, given two i-vectors $\boldsymbol{\phi}_i$ and $\boldsymbol{\phi}_j$, the LLR score for the PLDA model is given by

$$\begin{aligned} s_{ij} &= \boldsymbol{\phi}_i^T \boldsymbol{\Lambda} \boldsymbol{\phi}_j + \boldsymbol{\phi}_j^T \boldsymbol{\Lambda} \boldsymbol{\phi}_i + \boldsymbol{\phi}_i^T \boldsymbol{\Gamma} \boldsymbol{\phi}_i + \boldsymbol{\phi}_j^T \boldsymbol{\Gamma} \boldsymbol{\phi}_j \\ &+ (\boldsymbol{\phi}_i + \boldsymbol{\phi}_j)^T \boldsymbol{c} + k, \end{aligned} \tag{1}$$

where the parameters $\boldsymbol{\Lambda}$, $\boldsymbol{\Gamma}$, $\boldsymbol{c}$ and k can be calculated from the parameters of the PLDA model (see [13] for details). The idea of DPLDA is to train $\boldsymbol{\Lambda}$, $\boldsymbol{\Gamma}$, $\boldsymbol{c}$ and k directly for the speaker verification task, i.e., given two i-vectors, tell whether they are from the same speaker or not. This is achieved by forming trials (usually all possible) from the training data and optimizing, e.g., the binary cross-entropy or the SVM objective. In this work we use the binary cross-entropy objective.

Normally, DPLDA is trained iteratively using full batches, i.e., each update of the model parameters is calculated based on all training data. Whenever the DPLDA model is trained individually, we train it in this way. However, for an end-to-end system this would require too much memory and computational time. As is common for neural networks, we therefore calculate each update of the model parameters based on a minibatch, i.e., a randomly selected subset of the training data. Contrary to the individual training of **f2s** and **s2i**, we use the ADAM optimizer [21] since it may be more robust to different learning rate requirements of the differnt modules compared to standard SGD. We half the learning rate whenever we see no improvment in C_{min}^{Prm} on the development set after an epoch (defined to be 200 batches).

Due to the fact that the training trials are formed by combining training utterances, it is not obvious how to optimally select the data for minibatches. In our experiments, we use the following procedure:

1. For each speaker, randomly group his/her utterances into pairs.[2]
2. For each minibatch, randomly select (without replacement) N pairs and use all trials that can be formed from the corresponding utterances. If the last pair is selected, repeat Step 1.

3.4. End-to-end system

After the individual components described in the previous subsections have been trained individually, they are combined to an end-to-end system. Unfortunately, combining the modules as they are leads to large memory requirements of the end-to-end system. This happens mainly for two reasons. First, contrary to the individual training of the modules, the PCA projection now needs to be part of the network in order for the **f2s** and **s2i** modules to be connected. If the total memory on the GPU is 4GB, then the PCA matrix with 122880×4000 parameters uses about half of the available memory. Second, the **f2s** now needs to process all frames from many different utterances in one batch to obtain the sufficient number of trials for the DPLDA module. If the three modules are combined as they are, we can use only approximately 2 utterances per minibatch which is not sufficient for effective training (see Section 5.3 for more discussion about this).

To mitigate the problem of the large PCA matrix we will, before doing the complete end-to-end training, train only the **s2i** NN and the DPLDA model jointly. As for the individual training of **s2i**, we can use pre-calculated input that includes the PCA projection since this input is fixed as long as **f2s** is not updated. To mitigate the large memory requirements of the **f2s** module, we modify the training procedure to keep less intermediate results in memory. Specifically, in usual NN training, the input is first *forward propagated* through the network to get the output of each layer. These outputs are stored in memory and used during *backpropagation* to obtain the derivative of the loss with respect to each model parameter. For the part of **f2s** that calculates responsibilities (Block A in Figure 1), this results in $N(1500 + 1500 + 1500 + 1500 + 2048)$ variables to store in memory, where N is the total number of frames. This is much more than in subsequent modules (after pooling the frames into sufficient statistics, $\boldsymbol{F}$ and $\boldsymbol{N}$) because in this part of the network the layer outputs are per frame whereas in subsequent modules

[2] If a speaker has only one utterance, this utterance will be used as a "pair". If a speaker has another uneven number of utterances, one of the pairs will be given three utterances.

the layer outputs correspond to the whole utterance. Thus, in order to reduce the memory usage, we calculate the sufficient statistics for one utterance at the time and discard all the layer outputs from Block A once the sufficient statistics for the utterance have been calculated. When the sufficient statisics for all utterances have been obtained, we continue the forward propagation in the normal way, keeping all outputs in memory. During backpropagation, we recalculate the outputs when needed. This is achieved in a similar way as in `scan_checkpoints`[3]. This trick allows us to use minibatches of 10 pairs (N) instead of approximately 2.

4. EXPERIMENTAL RESULTS

We report results in EER as well as in the average minDCF for the probability of target trials equal to 0.01 and 0.005 (C_{min}^{Prm}). These were the two operating points of interest in the NIST SRE'16 [18]. Table 1 shows the results for the two baselines, the end-to-end system as well as systems where only some stages of the baseline have been replaced by a NN. Looking at the first and second row, we can see the DPLDA performs better than generatively trained PLDA on all sets. This is consistent with our previous findings on NIST SRE'16 [16]. The third and fourth row show the performance when either the UBM or the i-vector extractor is replaced by a NN but the other parts remain the same. We can see that the **f2s** NN can well mimic the GMM UBM whereas replacing the i-vector extractor with the **s2i** NN degrades the results. Notice that these results are only for the generatively trained PLDA model. The fifth row shows the results when we train a **s2i** module on the output from the **f2s** module instead of the standard sufficient statistics. The result is similar to the **s2i** being trained on standard statistics. Interestingly, when we further change from generative trained PLDA to DPLDA, the model performs better than both baselines. This suggests that the output from the **s2i** can well discriminate between speakers but may not well fulfill the PLDA model assumptions. After individual training of all blocks, we proceed with joint training of the **f2s** and **s2i** modules, using L2 regularization (tuned on the **dev** set) towards the parameters of the initial models. For this we use a batch size (N in Section 3.4) of 5000 pairs. As can be seen in the seventh row of Table 1, the joint training of the two modules improves the performance on **short lang** and on **PRISM lang** but not on **SRE16**. Finally, the last row shows the performance when all modules are trained jointly. For this training, we can only use $N = 10$ as discussed in Section 3.3. As can be seen, the performance is almost unchanged from the previous row. We believe larger minibatches or a better training scheme is needed for this step to work better. This will be studied in future work. In summary, the final system outperformed the DPLDA baseline with 2.2%, 4.3% 19.7% in C_{min}^{Prm} on *SRE16*, *short lang* and *PRISM lang* respectively. In EER, the improvements were 9.1%, 8.1%, and 9.7%.

5. ANALYSIS AND DISCUSSION

5.1. Architectures for f2s network

We have performed several experiments in order to choose optimal architecture for the **f2s** module. As discussed in Section 3.1, we trained only the part of the module that predicts a vector of responsibilities for each frame. The last layer of the whole **f2s** network is kept fixed for all of our experiments. Table 2 presents the results for some of architectures we tried for the trainable part of the **f2s** unit. In this subnetwork, we varied the number of hidden layers as well as experimented with adding contextual information to the input of the network. The results indicate that all of the tested architectures can successfully mimic the original UBM statistic extraction and even provide slight performance improvement compared to the baseline system. However, for the final end-to-end system we decided to use the most complex architecture out of all tried; it corresponds to the second line of the Table 2. Even though the **f2s** module does not seem to take any advantage from more complex architecture, we believe that the end-to-end network could utilize the contextual information and complex dependencies in the first module to improve overall performance.

[3]http://www.deeplearning.net/software/theano/library/scan.html

5.2. Architectures for s2i network

We carried out several experiments to find the optimal architecture for the **s2i** module. All experiments presented in this section use two 600 dimensional hidden layers, as this was found to provide optimal performance. The output layer has either 250 or 600 dimensions depending on whether the reference i-vectors have been reduced by LDA or not.

We first analyzed the effect of reducing the MAP adapted supervector to different dimensionalities. Dimensionalities from 4000 to 8000 provided comparable performance, whereas reducing the supervector dimension below 4000 harmed the performance, specially in terms of C_{min}^{Prm}. Therefore, we reduced the supervector dimensionality to 4000 in the remaining experiments.

Next, we explored different preprocessing techniques of the reference i-vectors in combination with different output layers of the module. As preprocessing techniques, we explored length-normalization and within-class-covariance normalization (WCCN). As output layers, we compared whether a length-normalization (LN) output layer was preferable to a linear output layer (LO). When performing these experiments, a linear output layer had to be used first for around 5 iterations and then switched to the LN layer in order to allow convergence. The results are presented in Table 3. We can see that using the LN output layer enhances performance. Further, we see that preprocessing the reference i-vectors with WCCN improves the performance. In the remaining experiments we use WCNN+LN for i-vector preprocessing and a LN output layer.

In the final set of experiments we analyze the system performance when using different training objectives: **1)** Minimizing the mean square error between reference i-vectors and the ones produced by the NN. **2)** Minimizing cosine distance between both sets of i-vectors. **3)** Maximizing the PLDA scores obtained by scoring reference and produced vectors with a PLDA system trained on the reference i-vectors. We also experimented with reference i-vectors that were reduced by LDA (to 250 dimensions) before being length-normalized as well as with different regularization methods. The results for these experiments are shown in Table 4. It is clear that using LDA for preprocessing reference i-vectors is helpful. The differences between the different objective functions were marginal. Between regularization methods, L1 provided better overall gains. In the end-to-end system we decided to use the cosine distance objective with LDA and L1 regularization. For simplicity, we did not use the PLDA objective.

5.3. Minibatch design and sizes

As mentioned in Section 3.3, it is not obvious how to optimally select the data for minibatches. Contrary to typical NN training scenarios, the training data in the speaker verification scenario are statistically

Table 1. Overall results, C_{min}^{Prm} and EER. Modules marked with a '*' are trained jointly. Other modules are trained sequentially.

				SRE16		short lang		PRISM lang	
System Name	stats	i-vector	PLDA	C_{min}^{Prm}	EER	C_{min}^{Prm}	EER	C_{min}^{Prm}	EER
Baseline	UBM	i-extractor	Gen.	0.988	17.645	0.699	10.303	0.411	3.902
Baseline DPLDA	UBM	i-extractor	Discr.	0.975	16.902	0.616	9.462	0.360	3.461
f2s	NN	i-extractor	Gen.	0.980	16.809	0.687	9.866	0.394	3.713
s2i	UBM	NN	Gen.	0.988	16.686	0.788	11.141	0.430	4.584
f2s-s2i	NN	NN	Gen.	0.982	16.226	0.780	11.523	0.432	4.616
f2s-s2i-DPLDA	NN	NN	Discr	0.953	15.091	0.597	9.328	0.300	3.426
s2i-DPLDA_joint	NN	NN*	Discr*	0.936	15.166	0.586	8.599	0.287	3.123
f2s-s2i-DPLDA_joint	NN*	NN*	Discr*	0.936	15.170	0.587	8.661	0.287	3.125

Table 2. Architectures for **f2s** . Results are on NIST SRE'10, cond. 5, females, C_{min}^{Prm} and EER. The numbers in the *architecture* refers to input dimension, #layers $\times$ layer size and output dimension.

Architecture	EER	C_{min}^{Prm}
Baseline (no NN)	2.37	0.270
NN (360_4×1500_2048)	2.17	0.253
NN (360_2×1500_2048)	2.20	0.254
NN (60_2×1500_2048)	2.27	0.268

Table 3. Combinations of different reference i-vector preprocessing and output layers for **s2i** module. Results on NIST SRE'10, cond. 5, females, C_{min}^{Prm} and EER. LO $\rightarrow$ LN indicates that the model was initially trained with LO, then with LN.

ivec prep.	Output Layer	EER	C_{min}^{Prm}
Baseline (no NN)	-	2.40	0.294
LN	LO	2.86	0.318
LN	LO $\rightarrow$ LN	2.82	0.302
WCCN + LN	LO	2.76	0.299
WCCN + LN	LO $\rightarrow$ LN	2.59	0.292

Table 4. Training objectives for **s2i** module. Results on NIST SRE'10, cond. 5, females, C_{min}^{Prm} and EER.

Obj. Function	LDA	REG	EER	C_{min}^{Prm}
Mean square error	NO	-	2.59	2.92
Mean square error	YES	-	2.57	2.83
Cosine distance	NO	-	2.56	2.90
Cosine distance	YES	-	2.55	2.84
Cosine distance	YES	L1	2.43	2.81
Cosine distance	YES	L2	2.50	2.84
PLDA	YES	L1	2.36	2.84

dependent due to the fact that the training trials are formed by combining training utterances [22]. Statistically dependent trials provide less information about the optimal model parameters than independent ones, i.e., they are less useful for training. In this aspect, minibatches should therefore ideally consist of independent trials. However, two more aspects need to be considered. First, selecting a set of trials to form a minibatch is not computationally efficient, instead we should select a set of utterances and use all possible trials. For example, instead of using (A-B) and (C-D) in a minibatch, where (X-Y) indicates a trial from the utterances X and Y, we should use all the trials (A-B), (A-C), (A-D), (B-C), (B-D), (C-D). This is because all the utterances A, B, C, and D need to be propagated through the **f2s** and **s2i** module anyway, and because the DPLDA model can very efficiently utilize all possible trials from a set of i-vectors [14].

The second aspect to consider is that target and non-target trials should be reasonably balanced in the minibatch. The method we used in the experiments (described in Section 3.3) results in minibatches that have few target trials per minibatch. An approach that results in more target trials per minibatch is to let the minibatches consist of all utterances from a few number of speaker. Although this results in more target trials, these trials are statistically dependent since the same speaker and utterances are used in several of the trials. Our initial experiments suggest that this approach is worse than the one we used in this work.

In Table 5, we present an analysis on how the training is affected by the minibatch size. We consider two metrics. First, C_{min}^{Prm} on the **dev** set. Second, the training loss on the whole training set for the final model. The results are given for the case of L2 regularization 1.0 (the optimal) and 0.1. We can see that the minibatch size needs to be several thousand in order for the resulting L to be comparable to the L obtained by training with full batches (LBFGS). This holds especially for the weaker regularization. Fortunately, C_{min}^{Prm} (on the **dev** set) converges faster than L which suggest that even though we do not manage to optimize the model as well as with LBFGS, it is sufficient from the performance point of view. However, this may not be the case for more complicated models or data sets. In these experiments we checked L after each epoch (defined to be 20 batches, i.e., less than in the end-to-end experiments) and halved the learning rate if it did not improve. The reason we used L as halving criteria instead of C_{min}^{Prm} was that these experiments were intended to show how well minibatch training can optimize the model compared to using full batches.

6. CONCLUSIONS

In this work, we have developed an end-to-end speaker verification system that outperforms an i-vector+PLDA baseline on three different datasets with utterances from many different languages and of both long and short durations. The system was constrained to behave similar to an i-vector + PLDA system. In this way we mitigated overfitting which normally limits the performance of end-to-end systems. This was a conservative approach and future work should explore if less constrained system can perform better, in particular as complemt

Table 5. Effect of minibatch size. Results in C_{min}^{Prm} on development set and training lost of the final model on the training set, L.

Training method	Batch size	L2: 1.0 C_{min}^{Prm}	L2: 1.0 L	L2: 0.1 C_{min}^{Prm}	L2: 0.1 L
LBFGS	∼85k	0.554	0.203	0.566	0.128
ADAM	10	0.576	0.287	0.670	0.301
-”-	20	0.558	0.282	0.558	0.258
-”-	50	0.540	0.253	0.537	0.236
-”-	100	0.546	0.244	0.549	0.206
-”-	500	0.544	0.231	0.568	0.173
-”-	1000	0.546	0.225	0.562	0.160
-”-	2500	0.552	0.217	0.568	0.146
-”-	5000	0.557	0.213	0.565	0.136

to i-vector+PLDA systems. We found that training all three modules of the system jointly is difficult due to large memory requirements. However, joint training of two modules was effective. In future work we therefore want to develop more effective strategies for joint training of all three modules. The proposed system is designed for using single enrollment sessions, and extending it to deal with multiple enrollment sessions is an important future work.

7. REFERENCES

[1] G. E. Dahl, D. Yu, L. Deng, and A. Acero, “Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 30–42, Jan 2012.

[2] F. Schroff, D. Kalenichenko, and J. Philbin, “Facenet: A unified embedding for face recognition and clustering,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[3] Alicia Lozano-Diez, Anna Silnova, Pavel Matějka, Ondřej Glembek, Oldřich Plchot, Jan Pešán, Lukáš Burget, and Joaquin Gonzalez-Rodriguez, “Analysis and optimization of bottleneck features for speaker recognition,” in *Proceedings of Odyssey 2016*. 2016, vol. 2016, pp. 352–357, International Speech Communication Association.

[4] Y. Lei, N. Scheffer, L. Ferrer, and M. McLaren, “A novel scheme for speaker recognition using a phonetically-aware deep neural network,” in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 1695–1699.

[5] S. Novoselov, T. Pekhovsky, O. Kudashev, V. S. Mendelev, and A. Prudnikov, “Non-linear plda for i-vector speaker verification,” in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Sept 2015, pp. 214–218.

[6] Gautam Bhattacharya, Md. Jahangir Alam, Patrick Kenny, and Vishwa Gupta, “Modelling speaker and channel variability using deep neural networks for robust speaker verification,” in *2016 IEEE Spoken Language Technology Workshop, SLT 2016, San Diego, CA, USA, December 13-16*, 2016.

[7] O. Ghahabi and J. Hernando, “Deep belief networks for i-vector based speaker recognition,” in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 1700–1704.

[8] E. Variani, X. Lei, E. McDermott, I. L. Moreno, and J. Gonzalez-Dominguez, “Deep neural networks for small footprint text-dependent speaker verification,” in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 4052–4056.

[9] G. Heigold, I. Moreno, S. Bengio, and N. Shazeer, “End-to-end text-dependent speaker verification,” in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2016, pp. 5115–5119.

[10] D. Snyder, P. Ghahremani, D. Povey, D. Garcia-Romero, Y. Carmiel, and S. Khudanpur, “Deep neural network-based speaker embeddings for end-to-end speaker verification,” in *2016 IEEE Spoken Language Technology Workshop (SLT)*, Dec 2016, pp. 165–170.

[11] David Snyder, Daniel Garcia-Romero, Daniel Povey, and Sanjeev Khudanpur, “Deep neural network embeddings for text-independent speaker verification,” in *Interspeech 2017*, Aug 2017.

[12] S. X. Zhang, Z. Chen, Y. Zhao, J. Li, and Y. Gong, “End-to-end attention based text-dependent speaker verification,” in *2016 IEEE Spoken Language Technology Workshop (SLT)*, Dec 2016, pp. 171–178.

[13] L. Burget, O. Plchot, S. Cumani, O. Glembek, P. Matějka, and N. Brümmer, “Discriminatively trained probabilistic linear discriminant analysis for speaker verification,” in *Proc. of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Prague, CZ, May 2011.

[14] S. Cumani, N. Brümmer, L. Burget, P. Laface, O. Plchot, and V. Vasilakis, “Pairwise discriminative speaker verification in the i–vector space,” *IEEE Transactions on Audio, Speech and Language Processing*, vol. 21, no. 6, pp. 1217–1227, june 2013.

[15] L. Ferrer, H. Bratt, L. Burget, H. Cernocky, O. Glembek, M. Graciarena, A. Lawson, Y. Lei, P. Matejka, O. Plchot, et al., “Promoting robustness for speaker modeling in the community: the prism evaluation set,” *https://code.google.com/p/prism-set/*, 2012.

[16] Oldřich Plchot, Pavel Matějka, Anna Silnova, Ondřej Novotný, Mireia Diez, Johan Rohdin, Ondřej Glembek, Niko Brümmer, Albert Swart, Jesús Jorrín-Prieto, Paola García, Luis Buera, Patrick Kenny, Jahangir Alam, and Gautam Bhattacharya, “Analysis and Description of ABC Submission to NIST SRE 2016,” in *Interspeech 2017*, Stockholm, Sweden, 2017.

[17] 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. PP, no. 99, 2010.

[18] “The 2016 NIST speaker recognition evaluation plan (sre16),” https://www.nist.gov/file/325336.

[19] M. Karafiát, F. Grézl, K. Veselý, M. Hannemann, I. Szőke, and J. Černocký, “But 2014 babel system: Analysis of adaptation in nn based systems,” in *Proceedings of Interspeech 2014*. 2014, pp. 3002–3006, International Speech Communication Association.

[20] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, “Speaker verification using adapted gaussian mixture models,” *Digital Signal Processing*, vol. 10, pp. 19–41, 2000.

[21] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *CoRR*, vol. abs/1412.6980, 2014.

[22] J. Rohdin, S. Biswas, and K. Shinoda, “Robust discriminative training against data insufficiency in plda-based speaker verification,” *Computer Speech & Language*, vol. 35, pp. 32 – 57, 2016.