End-to-End Text-Dependent Speaker Verification

Georg Heigold[*]

Saarland University & DFKI, Germany
georg.heigold@dfki.de

Ignacio Moreno, Samy Bengio, Noam Shazeer

Google Inc., USA
{elnota,bengio,noam}@google.com

Abstract

In this paper we present a data-driven, integrated approach to speaker verification, which maps a test utterance and a few reference utterances directly to a single score for verification and jointly optimizes the system's components using the same evaluation protocol and metric as at test time. Such an approach will result in simple and efficient systems, requiring little domain-specific knowledge and making few model assumptions. We implement the idea by formulating the problem as a single neural network architecture, including the estimation of a speaker model on only a few utterances, and evaluate it on our internal "Ok Google" benchmark for text-dependent speaker verification. The proposed approach appears to be very effective for big data applications like ours that require highly accurate, easy-to-maintain systems with a small footprint.

Index Terms: speaker verification, end-to-end training, deep learning.

1. Introduction

Speaker verification is the process of verifying, based on a speaker's known utterances, whether an utterance belongs to the speaker. When the lexicon of the spoken utterances is constrained to a single word or phrase across all users, the process is referred to as global password text-dependent speaker verification. By constraining the lexicon, text-dependent speaker verification aims to compensate for phonetic variability, which poses a significant challenge in speaker verification [1]. At Google, we are interested in text-dependent speaker verification with the global password "Ok Google." The choice of this particularly short, approximately 0.6 seconds long global password relates to the Google Keyword Spotting system [2] and Google VoiceSearch [3] and facilitates the combination of the systems.

In this paper, we propose to directly map a test utterance together with a few utterances to build the speaker model, to a single score for verification. All the components are jointly optimized using a verification-based loss following the standard speaker verification protocol. Compared to existing approaches, such an end-to-end approach may have several advantages, including the *direct modeling* from utterances, which allows for capturing long-range context and reduces the complexity (one vs. number of frames evaluations per utterance), and the *direct and joint estimation*, which can lead to better and more compact models. Moreover, this approach often results in considerably simplified systems requiring fewer concepts and heuristics.

More specifically, the contributions of this paper include:

- formulation of end-to-end speaker verification architecture, including the estimation of a speaker model on a few utterances (Section 4);
- empirical evaluation of end-to-end speaker verification, including comparison of frame (i-vectors, d-vectors) and utterance-level representations (Section 5.2) and analysis of the end-to-end loss (Section 5.3);
- empirical comparison of feedforward and recurrent neural networks (Section 5.4).

This paper focuses on text-dependent speaker verification for small footprint systems, as discussed in [4]. But the approach is more general and could be used similarly for text-independent speaker verification.

In previous studies, the verification problem is broken down into more tractable, but loosely connected subproblems. For example, the combination of i-vector and probabilistic linear discriminant analysis (PLDA) [5, 6] has become the dominant approach, both for text-independent speaker verification [7, 8, 5, 6] and text-dependent speaker verification [9, 10, 11]. Hybrid approaches that include deep learning based components have also proved to be beneficial for text-independent speaker recognition [12, 13, 14]. For small footprint systems, however, a more direct deep learning modeling may be an attractive alternative [15, 4]. To the best of our knowledge, recurrent neural networks have been applied to related problems such as speaker identification [16] and language identification [17], but not to the speaker verification task. The proposed neural network architecture can be thought of as joint optimization of a generative-discriminative hybrid and is in the same spirit as deep unfolding [18] for adaptation.

The remainder of the paper is organized as follows. Section 2 provides a brief overview of speaker verification in general. Section 3 describes the d-vector approach. Section 4 introduces the proposed end-to-end approach to speaker verification. An experimental evaluation and analysis can be found in Section 5. The paper is concluded in Section 6.

2. Speaker Verification Protocol

The standard verification protocol can be divided into the three steps: training, enrollment, and evaluation, which we describe in more detail next.

Training In the training stage, we find a suitable internal speaker representation from the utterance, allowing for a simple scoring function. In general, this representation depends on the type of the model (e.g., Gaussian subspace model or deep neural network), the representation level (e.g., frame or utterance), and the model training loss (e.g., maximum likelihood or softmax). State-of-the art representations are a summary of frame-level information, such as i-vectors [7, 8] and d-vectors (Section 3).

Enrollment In the enrollment stage, a speaker provides a few utterances (see Table 1), which are used to estimate a speaker model. A common choice is to average the i-vectors [19] or d-vectors [15, 4] of these utterances.

[*] Work done while the author was at Google.

Evaluation During the evaluation stage, the verification task is performed and the system is evaluated. For verification, the value of a scoring function of the utterance X and the test speaker spk, $S(X, spk)$, is compared against a pre-defined threshold. We accept if the score exceeds the threshold, i.e., the utterance X comes from speaker spk, and reject otherwise. In this setup, two types of error can occur: false reject and false accept. Clearly, the false reject rate and the false accept rate depend on the threshold. When the two rates are equal, the common value is called equal error rate (EER).

A simple scoring function is the cosine similarity between the speaker representation $f(X)$ of an evaluation utterance X (see paragraph "Training") and the speaker model m_{spk} (see paragraph "Enrollment"):

$$S(X, spk) = [f(X)^{\top} m_{spk}]/[\|f(X)\| \, \|m_{spk}\|].$$

PLDA has been proposed as a more refined, data-driven scoring approach.

3. D-Vector Baseline Approach

D-vectors are derived from a deep neural network (DNN), as speaker representation of an utterance. A DNN consists of the successive application of several non-linear functions in order to transform the speaker utterance into a vector where a decision can be easily made. Fig. 1 depicts the topology of our baseline DNN. It includes a locally-connected layer [4] and several fully connected layers. All layers use ReLU activation except the last, which is linear. During the training stage, the parameters of the DNN are optimized using the softmax loss, which, for convenience, we define to comprise a linear transformation with weight vectors w_{spk} and biases b_{spk}, followed by the softmax function and the cross-entropy loss:

$$l_{\text{softmax}} = -\log \frac{\exp(w_{spk}^{\top} y + b_{spk})}{\sum_{\tilde{spk}} \exp(w_{\tilde{spk}}^{\top} y + b_{\tilde{spk}})}$$

where the activation vector of the last hidden layer is denoted by y and spk denotes the correct speaker. The normalization is over all competing training speakers $\tilde{spk}$.

After the training stage is completed, the parameters of the DNN are fixed. Utterance d-vectors are obtained by averaging the activation vectors of the last hidden layer for all frames of an utterance. Each utterance generates one d-vector. For enrollment, the speaker model is given by the average over the d-vectors of the enrollment utterances. Finally, during the evaluation stage, the scoring function is the cosine similarity between the speaker model d-vector and the d-vector of a test utterance.

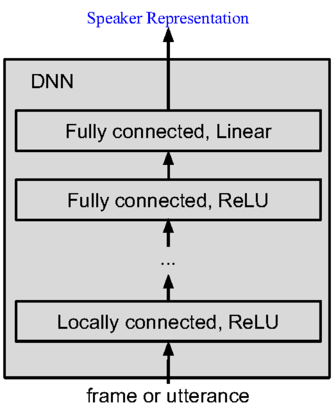

Figure 1: *Deep neural network (DNN) with a locally-connected layer followed by fully-connected layers.*

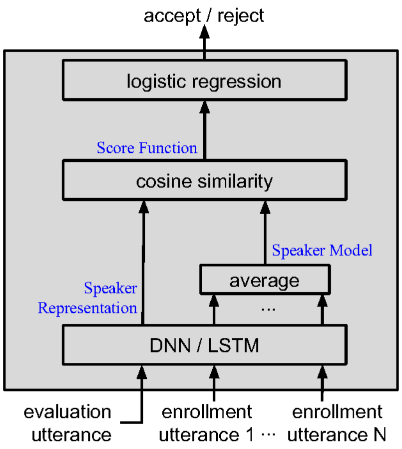

Figure 2: *End-to-end architecture: the input is an "evaluation" utterance and up to N "enrollment" utterances, which the network maps to a single output node (accept/reject). The "enrollment" utterances are used to estimate the speaker model.*

Criticism about this baseline approach includes the limited context of the d-vectors derived from (a window of) frames and the type of the loss. The softmax loss attempts to discriminate between the true speaker and all competing speakers but does not follow the standard verification protocol in Section 2. As a result, heuristics and scoring normalization techniques becomes necessary to compensate for inconsistencies. Moreover, the softmax loss does not scale well with more data as the computational complexity is linear in the number of training speakers and requires a minimum amount of data per speaker to estimate the speaker-specific weights and biases. The complexity issue (but not the estimation issue) can be alleviated by candidate sampling [20].

Similar concerns can be expressed over the alternative speaker verification approaches, where some of the component blocks are either loosely connected or not directly optimized following the speaker verification protocol. For example, GMM-UBM [7] or i-vector models does not directly optimize a verification problem; the PLDA [5] model is not followed a refinement of the i-vector extraction; or long contextual features may be ignored by frame-based GMM-UBM models [7].

4. End-To-End Speaker Verification

In this section, we integrate the steps of the speaker verification protocol (Section 2) into a single network, see Fig. 2. The input of this network consists of an "evaluation" utterance and a small set of "enrollment" utterances. The output is a single node indicating accept or reject. We jointly optimized this end-to-end architecture using DistBelief [21], a predecessor of TensorFlow [22]. In both tools, complex computational graphs such as the one defined by our end-to-end topology can be decomposed into a sequence of operations with simple gradients such as sums, divisions, and cross products of vectors. After the training step, all network weights are kept fixed, except for the bias of the one-dimensional logistic regression (Fig. 2) which is manually tuned on the enrollment data. Apart from this, nothing is done in the enrollment step as the speaker model estimation is part of the network. At test time, we feed an evaluation utterance and the enrollment utterances of a speaker to be tested to the network, which directly outputs the decision.

We use neural networks to obtain the speaker representation

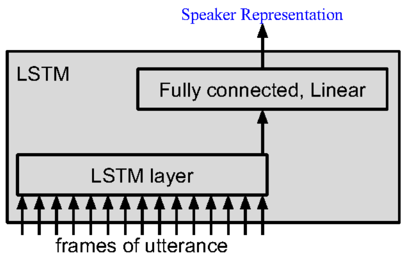

Figure 3: *Long short-term memory recurrent neural network (LSTM) with a single output.*

of an utterance. The two types of networks we use in this work are depicted in Figs. 1 and 3: a deep neural network (DNN) with locally-connected and fully connected layers as our baseline DNN in Section 3 and a long short-term memory recurrent neural network (LSTM) [23, 24] with a single output. DNNs assume a fixed-length input. To comply with this constraint, we stack the frames of a sufficiently large window of fixed length over the utterance and use them as the input. This trick is not needed for LSTMs but we use the same window of frames for better comparability. Unlike vanilla LSTMs which have multiple outputs, we only connect the last output to the loss to obtain a single, utterance-level speaker representation.

The speaker model is the average over a small number of "enrollment" representations (Section 2). We use the same network to compute the internal representations of the "test" utterance and of the utterances for the speaker model. The actual number of utterances per speaker available in training typically is much larger (a few hundred or more) than in enrollment (fewer than ten), see Table 1. To avoid a mismatch, we sample for each training utterance only a few utterances from the same speaker to build the speaker model at training time. In general, we cannot assume to have N utterances per speaker. To allow for a variable number of utterances, we pass a weight along with the utterance to indicate whether to use the utterance.

Finally, we compute the cosine similarity between the speaker representation and the speaker model, $S(X, spk)$, and feed it to a logistic regression including a linear layer with a bias. The architecture is optimized using the end-to-end loss

$$l_{e2e} = -\log p(target) \tag{1}$$

with the binary variable $target \in \{\text{accept}, \text{reject}\}$, $p(\text{accept}) = (1+\exp(-wS(X, spk)-b))^{-1}$, and $p(\text{reject}) = 1-p(\text{accept})$. The value $-b/w$ corresponds with the verification threshold.

The input of the end-to-end architecture are $1 + N$ utterances, i.e., an utterance to be tested and up to N different utterances of the same speaker to estimate the speaker model. To achieve a good tradeoff between data shuffling and memory, the input layer maintains a pool of utterances to sample $1+N$ utterances from for each training step and gets refreshed frequently for better data shuffling. As a certain number of utterances of the same speaker is needed for the speaker model, the data is presented in small groups of utterances of the same speaker.

5. Experimental Evaluation

We evaluate the proposed end-to-end approach on our internal "Ok Google" benchmark.

5.1. Data Sets & Basic Setup

We tested the proposed end-to-end approach on a set "Ok Google" utterances collected from anonymized voice search logs. For improved noise robustness, we perform multistyle training. The data were augmented by artificially adding in car and cafeteria noise at various SNRs, and simulating different distances between the speaker and the microphone, see [2] for further details. Enrollment and evaluation data include only real data. Table 1 shows some data set statistics.

Table 1: Data set statistics.

	#utterances (#augmented)	#speakers	#utts / spk
train_2M	2M (9M)	4k	>500
train_22M	22M (73M)	80k	>150
enrollment	18k	3k	1-9
evaluation	20k	3k	3-5

The utterances are forced aligned to obtain the "Ok Google" snippets. The average length of these snippets is around 80 frames, for a frame rate of 100 Hz. Based on this observation, we extracted the last 80 frames from each snippet, possibly padding or truncating frames at the beginning of the snippet. The frames consist of 40 log-filterbanks (with some basic spectral subtraction) each.

For DNNs, we concatenate the 80 input frames, resulting in a 80x40-dimensional feature vector. Unless specified otherwise, the DNN consists of 4 hidden layers. All hidden layers in the DNN have 504 nodes and use ReLU activation except the last, which is linear. The patch size for the locally-connected layer of the DNN is 10×10. For LSTMs, we feed the 40-dimensional feature vectors frame by frame. We use a single LSTM layer with 504 nodes without a projection layer. The batch size is 32 for all experiments.

Results are reported in terms of equal error rate (EER), without and with t-norm score normalization [25].

5.2. Frame-Level vs. Utterance-Level Representation

First, we compare frame-level and utterance-level speaker representations, see Table 2. Here, we use a DNN as described in Fig. 1 with a softmax layer and trained on train_2M (Table 1) with 50% dropout [26] in the linear layer. The utterance-level approach outperforms the frame-level approach by 30%. Score normalization gives a substantial performance boost (up to 20% relative) in either case. For comparison, two i-vector baselines

Table 2: Equal error rates for frame-level and utterance-level speaker representations.

level	system	EER (%) raw	EER (%) t-norm
frame	i-vector [6]	5.77	5.11
	i-vector+PLDA [27]	4.66	4.89
	DNN, softmax [4]	3.86	3.32
utterance	DNN, softmax	2.90	2.28

are shown. The first baseline is based on [6], and uses 13 PLPs with first-order and second-order derivatives, 1024 Gaussians, and 300-dimensional i-vectors. The second baseline is based on [27] with 150 eigenvoices. The i-vector+PLDA baseline should be taken with a grain of salt as the PLDA model was only trained on a subset of the 2M_train data set (4k speakers and 50 utterances per speaker) due to limitations of our current implementation.[1] Also, this baseline does not include other refining techniques such as "uncertainty training" [10] that have been reported to give substantial additional gains under certain

[1] However, training with only 30 utterances per speaker gives almost the same results.

conditions. Note that compared to [15], we have improved our d-vectors significantly [4].

5.3. Softmax vs. End-to-End Loss

Next, we compare the softmax loss (Section 2) and end-to-end loss (Section 4) for training utterance-level speaker representations. Table 3 shows the equal error rates for the DNN in Fig. 1. If trained on the small training set (train_2M), the error rates on the raw scores are comparable for the different loss functions. While dropout gives a 1% absolute gain for softmax, we did not observe a gain from dropout for the end-to-end loss. Similarly, t-normalization helps by 20% for softmax, but not at all for the end-to-end loss. This result is in agreement with the degree of consistency between the training loss and the evaluation metric. In particular, the end-to-end approach assuming a global threshold in training (see Eq. (1)), can implicitly learn normalized scores that are invariant under different noise conditions etc. and makes score normalization redundant. When using the softmax DNN for initialization of the end-to-end training, the error rate is reduced from 2.86% to 2.25%, suggesting an estimation problem.

If trained on the larger training set (train_22M), the end-to-end loss clearly outperforms softmax, see Table 3. To reasonably scale the softmax layer to 80k speaker labels, we employed candidate sampling, similar to [20]. Again, t-normalization helps by 20% for softmax and softmax can catch up with the other losses, which do not benefit from t-normalization. The initialization for end-to-end training (random vs. "pre-trained" softmax DNN) does not make a difference in this case.

Although the step time for the end-to-end approach is larger than for softmax with candidate sampling because the speaker model is computed on the fly, the overall convergence times are comparable.

Table 3: Equal error rates for different losses, * is with candidate sampling.

loss	EER (%), raw / t-norm train_2M	train_22M
softmax	2.90 / 2.28	2.69 / 2.08*
end-to-end	2.86 / 2.85	2.04 / 2.14

The optimal choice of the number of utterances used to estimate the speaker model in training, referred to as the speaker model size, depends on the (average) number of enrollment utterances. In practice, however, smaller speaker model sizes may be more attractive to reduce the training time and make the training harder. Fig. 4 shows the dependency of the test equal error rate on the speaker model size, i.e., the number of utterances used to estimate the speaker model. There is a relatively broad optimum around a model size of 5 with 2.04% equal error rate, compared to 2.25% for a model size of 1. This model size is close to the true average model size, which is 6 for our enrollment set. Similar trends are observed for the other configurations in this paper (not shown). This indicates the consistency of the proposed training algorithm with the verification protocol and suggests that task-specific training tends to be better.

5.4. Feedforward vs. Recurrent Neural Networks

So far we focused on the "small footprint" DNN in Fig. 1 with one locally-connected and three fully-connected hidden layers. Next, we explore larger and different network architectures, regardless of their size and computational complexity. The results are summarized in Table 4. Compared to the small footprint DNN, the "best" DNN uses an additional hidden layer and gives

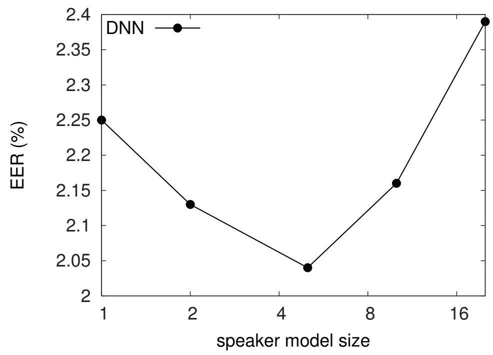

Figure 4: *Speaker model size vs. equal error rate (EER).*

a 10% relative gain. The LSTM in Fig. 3 adds another 30% gain over this best DNN. The number of parameters is comparable to that of the DNN but the LSTM involves about ten times more multiplications and additions. More hyperparameter tuning will hopefully bring the computational complexity further down to make it feasible. Slightly worse error rates are achieved with the softmax loss (using t-normalization, candidate sampling, dropout, and possibly early stopping, which were all not needed for the end-to-end approach). On train_2M, we observed similar relative gains in error rate over the respective DNN baselines.

Table 4: Equal error rates for different model architectures using end-to-end training, † is with t-norm score normalization and trained only on the smaller training set.

	EER (%)
frame-level DNN baseline	3.32†
DNN, "small footprint"	2.04
DNN, "best"	1.87
LSTM	1.36

6. Summary & Conclusion

We proposed a novel end-to-end approach to speaker verification, which directly maps the utterance to a score and jointly optimizes the internal speaker representation and the speaker model using the same loss for training and evaluation. Assuming sufficient training data, the proposed approach improved our best small footprint DNN baseline from over 3% to 2% equal error rate on our internal "Ok Google" benchmark. Most of the gain came from the utterance-level vs. frame-level modeling. Compared to other losses, the end-to-end loss achieved the same or slightly better results but with fewer additional concepts. In case of softmax, for example, we obtained comparable error rates only when using score normalization at runtime, candidate sampling to make training feasible, and dropout in training. Furthermore, we showed that the equal error rate can further be reduced to 1.4% using a recurrent neural network instead of a simple deep neural network, although at higher computational runtime cost. By comparison, a reasonable but not fully state-of-the-art i-vector/PLDA system gave 4.7%. Clearly, more comparative studies are needed. Nevertheless, we believe that our approach demonstrates a promising new direction for big data verification applications.

7. References

[1] H. Aronowitz, R. Hoory, J. W. Pelecanos, and D. Nahamoo, “New developments in voice biometrics for user authentication,” in *Interspeech*, Florence, Italy, Aug. 2011, pp. 17 – 20.

[2] R. Prabhavalkar, R. Alvarez, C. Parada, P. Nakkiran, and T. Sainath, “Automatic gain control and multi-style training for robust small-footprint keyword spotting with deep neural networks,” in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Brisbane, Australia, Apr. 2015, pp. 4704–4708.

[3] J. Schalkwyk, D. Beeferman, F. Beaufays, B. Byrne, C. Chelba, M. Cohen, M. Kamvar, and B. Strope, ““Your word is my command”: Google search by voice: A case study,” in *Advances in Speech Recognition: Mobile Environments, Call Centers and Clinics*. Springer, 2010, ch. 4, pp. 61–90.

[4] Y. Chen, I. Lopez-Moreno, and T. Sainath, “Locally-connected and convolutional neural networks for small footprint speaker recognition,” in *Interspeech*, Dresden, Germany, Sep. 2015.

[5] P. Kenny, “Bayesian speaker verification with heavy-tailed priors,” in *Proc. Odyssey Speaker and Language Recognition Workshop*, Brno, Czech Republic, Jul. 2010.

[6] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, “Front-end factor analysis for speaker verification,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2011.

[7] D. Reynolds, T. Quoter, and R. Dunn, “Speaker verification using adapted Gaussian mixture models,” *Digital Signal Processing*, vol. 10, no. 1, pp. 19–41, 2000.

[8] P. Kenny, G. Boulianne, P. Ouellet, and P. Dumouchel, “Joint factor analysis versus eigenchannels in speaker recognition,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, pp. 1435–1447, 2007.

[9] H. Aronowitz, “Text-dependent speaker verification using a small development set,” in *Proc. Odyssey Speaker and Language Recognition Workshop*, Singapore, Jun. 2012.

[10] T. Stafylakis, P. Kenny, P. Ouellet, P. Perez, J. Kockmann, and P. Dumouchel, “Text-dependent speaker recognition using PLDA with uncertainty propagation,” in *Interspeech*, Lyon, France, Aug. 2013.

[11] A. Larcher, K.-A. Lee, B. Ma, and H. Li, “Phonetically-constrained PLDA modeling for text-dependent speaker verification with multiple short utterances,” in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Vancouver, Canada, May 2013.

[12] D. Garcia-Romero, X. Zhang, A. McCree, and D. Povey, “Improving speaker recognition performance in the domain adaptation challenge using deep neural networks,” in *IEEE Spoken Language Technology Workshop (SLT)*, South Lake Tahoie, NV, USA, Dec. 2014, pp. 378–383.

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

[14] F. Richardson, D. Reynolds, and N. Dehak, “Deep neural network approaches to speaker and language recognition,” *IEEE Signal Processing Letters*, 2015.

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

[16] S. Parveen, A. Qadeer, and P. Green, “Speaker recognition with recurrent neural networks,” in *Interspeech*, Beijing, China, Oct 2000, pp. 16–20.

[17] J. Gonzalez-Dominguez, I. Lopez-Moreno, H. Sak, J. Gonzalez-Rodriguez, and P. Moreno, “Automatic language identification using long short-term memory recurrent neural networks,” in *Interspeech*, Singapore, Sep. 2014, pp. 2155–2159.

[18] J. Hershey, J. L. Roux, and F. Weninger, “Deep unfolding: Model-based inspiration of novel deep architectures,” *CoRR*, vol. abs/1409.2574, 2014.

[19] C. Greenberg, D. Bansé, G. Doddington, D. Garcia-Romero, J. Godfrey, T. Kinnunen, A. Martin, A. McCree, M. Przybocki, and D. Reynolds, “The NIST 2014 speaker recognition i-vector machine learning challenge,” in *Odyssey 2014: The Speaker and Language Recognition Workshop*, Joensuu, Finland, Jun. 2014.

[20] S. Jean, K. Cho, R. Memisevic, and Y. Bengio, “On using very large target vocabulary for neural machine translation,” *CoRR*, vol. abs/1412.2007, 2014.

[21] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, Q. Le, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Ng, “Large scale distributed deep networks,” in *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[22] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015, software available from tensorflow.org.

[23] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[24] H. Sak, A. Senior, and F. Beaufays, “Long short-term memory recurrent neural network architectures for large scale acoustic modeling,” in *Interspeech*, Singapore, Sep. 2014.

[25] R. Auckenthaler, M. Carey, and H. Lloyd-Thomas, “Score normalization for text-independent speaker verification systems,” *Digital Signal Processing*, vol. 10, no. 1-3, pp. 42–54, 2000.

[26] G. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Improving neural networks by preventing co-adaptation of feature detectors,” *CoRR*, vol. abs/1207.0580, 2012.

[27] D. Garcia-Romero and C. Espy-Wilson, “Analysis of i-vector length normalization in speaker recognition systems,” in *Interspeech*, Florence, Italy, Aug. 2011.