Interspeech 2018
2-6 September 2018, Hyderabad

An Investigation of Non-linear i-vectors for speaker verification

Nanxin Chen, Jesús Villalba, Najim Dehak

Center for Language and Speech Processing
Johns Hopkins University, Baltimore, MD
{bobchennan,jvillal7, ndehak3}@jhu.edu

Abstract

Speaker verification becomes increasingly important due to the popularity of speech assistants and smart home. *i-vectors* are used broadly for this topic, which use factor analysis to model the shift of average parameter in Gaussian Mixture Models. Recently by the progress of deep learning, high-level non-linearity improves results in many areas. In this paper we proposed a new framework of *i-vectors* which uses stochastic gradient descent to solve the problem of *i-vectors*. From our preliminary results stochastic gradient descent can get same performance as expectation-maximization algorithm. However, by back-propagation the assumption can be more flexible, so both linear and non-linear assumption is possible in our framework. From our result, both maximum a posteriori estimation and maximum likelihood lead to slightly better result than conventional *i-vectors* and both linear and non-linear system has similar performance.

Index Terms: speaker verification, generative model, stochastic gradient descent

1. Introduction

With the increasing popularity of smart phones and home assistants, speech related techniques draw more and more attention from both academic and industry. While speech recognition performance keeps improving, now people start to concern more about privacy issues. Given the fact that personal information is usually shared with those hardware, one natural question is that how we can make it secure. Speaker verification becomes a natural choice and becomes common on many home assistants. Speaker verification systems distinguish different speakers by their physiological characteristics, and reply to the corresponding user.

Speaker verification research has a very long history. Started from 20th century, Gaussian Mixture Model Universal Background Model (GMM-UBM) was proposed in [1] achieving great performance. The UBM model is trained on large corpus to model the speaker independent speech features distribution. Then, *Maximum a posteriori* is used to adapt the GMM parameters to any specific speaker. Later, joint factor analysis (JFA) [2, 3] enhanced the performance of GMM-UBM system. JFA makes the assumption that between and within speaker variability spaces are of low-rank. While JFA assumes different factors for channel and speaker, following research shows that even channel factors have information related to speaker and can improve overall performance. In 2011, Dehak et al. [4] proposed a simplified framework, which is broadly known as *i-vectors*, used single factor defined by a total variability matrix. The point estimate of the total variability factor (i-vector) is used as a new feature for other classifiers, e.g., probabilistic linear discriminant analysis (PLDA) [5] is used to make same/different speaker decisions[6, 7].

Recently, with the advent of deep learning technologies, more research [8, 9, 10, 11, 12, 13, 14, 15, 16] turned to use neural network to model the speaker variability. In [8, 9], neural network is used to replace GMM in GMM-UBM model to provide alignment and statistics. In [11], bottleneck features are extracted from neural network to make the GMM to get better unsupervised clustering of the feature space improving results. Later, different works are proposed to directly use the neural network for speaker verification [10, 12, 13, 14, 15, 16]. These networks require to be trained on large amount of data, usually using supervise learning, and show great performance on both text-dependent[10, 12, 13, 14, 15] and text-independent[16] speaker verification.

In this paper we proposed a new framework combining conventional *i-vectors* with neural network. More specifically, a generative model that generalizes *i-vectors* is proposed and we show that *i-vectors* is a special case of our new framework. Back-propagation based stochastic gradient descent is used to train the model parameters and infer the latent factors. While representing *i-vectors* in this new framework, we are enabled to add non-linearity to *i-vectors*. Our preliminary results on SRE16 show that

- minimum diverge estimation is helpful for inference
- Performance of linear and non-linear systems are similar
- Both maximum a posteriori and maximum likelihood can slightly improve results of conventional *i-vectors*

This paper is organized as follows. Section 2 introduces conventional i-vector systems and the proposed new i-vector framework. Section 3 reports result for i-vector baseline system and explores different factors: minimum diverge estimation, non-linearity, maximum likelihood estimation and *maximum a posteriori* estimation. Section 4 summarizes the paper and gives directions for future research.

2. i-Vectors and Non-linear i-vectors

2.1. i-vectors

The i-vector framework models the utterance features as a Gaussian mixture model (GMM). The shift of utterance dependent GMM supervector mean w.r.t. the universal background model is assumed to be

$$\mathbf{m} = \mathbf{M} + \mathbf{T}\mathbf{w} \tag{1}$$

where $\mathbf{M}$ is the original UBM mean, and $\mathbf{T}$ is a low-rank matrix mapping low-dimensional i-vector to the high-dimensional supervector space. The supervector is the concatenation of all mean parameters. $\mathbf{T}$ is usually called total variability matrix.

Expectation maximization is used to train both UBM and *i-vectors* model parameters. After extracting *i-vectors* as fixed-dimensional embedding, both cosine similarity and probabilistic linear discriminant analysis [5] can be used.

2.2. Non-linear i-vectors

2.2.1. Model definition

In this work as in the i-vector framework, we assumed that the utterance features are generated by a GMM,

$$\begin{aligned} \mathbf{z}_{ij} &\sim \text{Categorical}(\phi) \\ \mathbf{x}_{ij} &\sim \mathcal{N}(\mathbf{x}_{ij}|\mathbf{M}_{\mathbf{z}_{ij}} + G_{\mathbf{z}_{ij}}(\mathbf{w}_i;\theta), \mathbf{\Sigma}_{\mathbf{z_{ij}}}) \end{aligned} \tag{2}$$

where $\mathbf{x}_{ij}$ is frame j from utterance i, $\mathbf{z}_{ij}$ is one-hot vector of Gaussian occupations, and ϕ are the mixture weights. G is general function that maps the latent factor $\mathbf{w}$ to the shift between the utterance dependent supervector mean and UBM mean.

Conventional i-vectors can be considered as

$$G^{'}_{z_i}(\mathbf{w};\theta) = \mathbf{T}_{\theta\, z_i}\mathbf{w} \tag{3}$$

which is a special case of Equation (2). However, in this work we try to add non-linearity to the function G. One natural idea is to add activation function directly to the product of $\mathbf{T}_\theta$ and $\mathbf{w}$:

$$G^{*}_{z_i}(\mathbf{w};\theta) = g(\mathbf{T}_{\theta\, z_i}\mathbf{w}) \tag{4}$$

As in the i-vector framework, we will assume that $\mathbf{z}_{ij}$ posteriors are given by the GMM-UBM.

2.2.2. Theory

Theory behind proposed framework can be more general. We want to model connections between observed variables $\mathbf{X}$–the feature frames of a given utterance– and a latent variable $\mathbf{w}$. In a generative setting, we want to maximize the marginal likelihood of the observed data $\mathbf{X}$ given the model parameters θ, $\log p(\mathbf{X}|\theta)$. However in practice, this is intractable in most cases. Thus, we maximize a lower bound (ELBO) for that likelihood instead,

$$L(\mathbf{X},\theta,q) = \log p(\mathbf{X}|\theta) - D_{\text{KL}}(q(\mathbf{w}|\mathbf{X})||p(\mathbf{w}|\mathbf{X},\theta) \tag{5}$$

where q is any distribution over $\mathbf{w}$. Because KL-divergence is non-negative, $L(\mathbf{X},\theta,q)$ becomes a lower bound of $\log p(\mathbf{X}|\theta)$. It is actually easier to compute since it can be re-arranged as

$$L(\mathbf{X},\theta,q) = -E_{q(\mathbf{w}|\mathbf{X})}(\log q(\mathbf{w}|\mathbf{X}) - \log p(\mathbf{w},\mathbf{X}|\theta)) \tag{6}$$

Variational autoencoder is developed under this theory where encoder models $q(w|x)$ and decoder models $p(x|w)$. For embedding extraction we care more about $q(\mathbf{w}|\mathbf{X})$. In this work, we simplify this model and we just want to make a point estimate $\boldsymbol{\mu}$ of the latent factor. For that, we assume that the form of q is a Dirac $delta$ distribution,

$$q(\mathbf{w}|\mathbf{X}) = \delta(\mathbf{w}-\boldsymbol{\mu}) \tag{7}$$

where $\boldsymbol{\mu}$ is the latent factor point estimate. Thus, Equation (6) can be simplified as

$$\begin{aligned} L(\mathbf{w},\theta,q=\delta(\mathbf{w}-\boldsymbol{\mu})) &= H(q) + \log p(\mathbf{w}=\boldsymbol{\mu},\mathbf{X}|\theta) \\ &= \log p(\mathbf{X}|\mathbf{w}=\boldsymbol{\mu};\theta) + \log p(\mathbf{w}=\boldsymbol{\mu}) \end{aligned} \tag{8}$$

where $H(q)$ is the entropy of distribution q. So maximizing $L(\mathbf{X},\theta,q=\delta(\mathbf{w}-\boldsymbol{\mu}))$ derives *maximum a posteriori* inference [17].

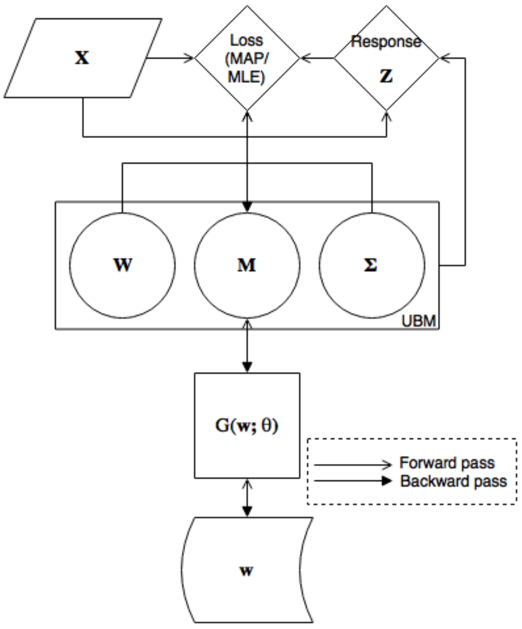

Figure 1: *Diagram of our model.* $\mathbf{W}$, $\mathbf{M}$, $\mathbf{V}$ *is weight, mean and co-variance of UBM, respectively. Response is fixed thus there is no backward pass from response to UBM.*

In our case, we care about how to find the optimal $\boldsymbol{\mu}$. Modeling the latent posterior $p(\mathbf{w}|\mathbf{X})$ is difficult, since $\mathbf{X}$ contains a variable number of frames, so we need to devise some kind of complex mechanism to accumulate the information of all the frames into the unique latent factor $\mathbf{w}$. However, similar to sparse coding, we treated this as an optimization problem, which simplified the problem. Instead of using closed form function to obtain $\boldsymbol{\mu}$, as it is proposed in the variational autoencoder framework [18], we proposed to assume that $\boldsymbol{\mu}$ is a trainable parameter that we obtain by back-propagation. In this manner, no encoder function is required but just the decoder distribution $p(\mathbf{X}|\mathbf{w})$. GMM-UBM is adopted as decoder in order to compare with *i-vectors*. The same UBM is trained on same datasets and used for both *i-vectors* and our method.

The diagram of whole process is given in Figure 1. In each mini-batch, we calculate response of each Gaussian based on the UBM model. Then all the embeddings are gathered according to data $\mathbf{X}$ in the mini-batch and update the supervector of UBM by function G. Loss value is calculated by data $\mathbf{X}$ and updated UBM. In backward, starting from loss value, gradient of θ and $\mathbf{w}$ is calculated by recursion. In each mini-batch, we updated θ and corresponding $\mathbf{w}$ in this mini-batch. In inference (i-vector extraction), only $\boldsymbol{\mu}$ is updated while θ is fixed.

Although our approach is minibatch-based training, we want to emphasize the similarities between EM algorithm and SGD. SGD in general can be seen as a special case of EM. In each mini-batch we optimize $\mathbf{w}$ given θ in previous batch and we optimize θ given last $\mathbf{w}$. Even each mini-batch cannot provide accurate gradient information and the step size is relatively small, with enough updates the system should converge to a similar point.

2.2.3. Loss function

In this work we consider two loss function according to Equation (8):

Corpus	#utts	Male	Female
SRE05	1165	38%	62%
SRE06	5245	45%	55%
SRE08	4504	39%	61%
swb1	30972	46%	54%

Table 1: *Training data details: number of utterances from each corpus and proportion of gender*

- Maximum a posteriori estimation. For the prior $p(w)$ we adopted standard normal distribution $\mathcal{N}(\mathbf{0},\mathbf{I})$.
- Maximum likelihood estimation where prior $p(w)$ has zero weight in Equation (8).

The models that we are using multiply the latent factor by a low-rank matrix $\mathbf{T}_\theta$, that creates an over-parametriced model. We can change to increase the scale of $\mathbf{T}_\theta$ and reduce the variance of $\mathbf{w}$ and vice-versa without any consequences in the objective function that we are using. To solve this and make the prior of $\mathbf{w}$ standard normal, we apply what we call minimum diverge estimation [2]. This essentially consists in transforming $\mathbf{T}_\theta$ as

$$\mathbf{T}_\theta = \mathbf{T}_\theta {\mathbf{K}_{\mathbf{ww}}}^{1/2} \tag{9}$$

where $\mathbf{K}_{\mathbf{ww}}$ is the co-variance matrix of $\mathbf{w}$. In diagonal case, ${\mathbf{K}_{\mathbf{ww}}}^{1/2}$ is the standard deviation of each dimension of $\mathbf{w}$. The intuition is to make $\mathbf{w}$ has standard normal distribution $\mathcal{N}(\mathbf{0},\mathbf{I})$ by normalizing $\mathbf{T}_\theta$. Experiments related to this is discussed in Section 3.3.1.

3. Experiments

3.1. Datasets

We experimented using the setup, that we used in our participation in NIST SRE16 evaluation [19, 20]. In this setup, the training set included two parts:

- swb1: Switchboard 1 data with each original cut segmented to a single 10 - 60 seconds. The duration is drawn form a uniform distribution
- sre568eng: 2005-2008 NIST English SRE data also uniformly segmented from 10 - 60 seconds.

There were 41859 utterances in total. More details of data is included in Table 1.

All frameworks were tested on Cantonese portion of the NIST SRE 2016 evaluation (SRE16). Enrollment utterances contained about 60 seconds of conversational telephone speech while the test utterances varied from 10 - 60 seconds. We estimated mean of *i-vectors* from NIST SRE16 development set "major" data and use it to center all *i-vectors* for enrollment and test data.

3.2. Experiments setup

UBM model was trained on whole training set and diagonal covariance was used in all experiments. Both UBM model and *i-vectors* baseline were trained using Kaldi [21].

Results for 400-dimensional *i-vectors* baseline with different number of diagonal Gaussians using MAP estimation is presented in Table 2. From the table, clearly more Gaussians give better result but the difference is not that large. Due to GPU memory limit we used the 256 Gaussians UBM to test our approach.

#Gaussians	EER	DCF10^{-2}	DCF10^{-3}
256	13.98	0.746	0.862
2048	**13.47**	**0.732**	**0.853**

Table 2: *400-dimensional i-vectors system baseline*

Model	Learning rate	EER	DCF10^{-2}	DCF10^{-3}
MDE	0.005	**13.18**	**0.722**	**0.858**
no MDE	0.005	14.05	0.748	0.878
no MDE	0.0005	13.21	0.723	0.858

Table 3: *Effect of minimum diverge estimation (MDE). With minimum diverge estimation fixed learning rate can be used. Otherwise adjusting learning rate manually is needed.*

In the following we introduce the method used to create the mini-batches that we used in our stochastic back-propagation algorithm. Because of the independence assumption, we can shuffle the frames in each utterance without altering the model. By shuffling the frames, we increase the variability of the feature chunks that we used to train the model. It is known that increasing randomness it is beneficial to make deep models to generalize well. After that we split each utterance into small chunks where each chunk included 128 frames. All those chunks were shuffled in each epoch before feeding them into our model. Each mini-batch contained 200 chunks. From our observations, both size of chunk and number of chunks in each mini-batch is quite important for final performance. The first one decides how accurate gradient of $\mathbf{w}$ is and the second one decides how much inter-session variability there is in the mini-batch. In the training learning rate is set to 0.001 and Adam[22] was used. For inference learning rate is set to 0.005 and 10 iterations are used to find the optimal $\mathbf{w}$. For fair comparison, we also set $\mathbf{w}$ to 400-dimensional embedding. For both *i-vectors* and non-linear *i-vectors*, probabilistic linear discriminant analysis (PLDA) was adopted as the back-end.

3.3. Results

3.3.1. Effect of minimum diverge estimation

As discussed in Section 2.2.3, minimum diverge estimation is used to reduce the effect of imbalance between scale of T_θ and variance of $\mathbf{w}$. Results are given in Table 3.

Clearly from table minimum diverge estimation is helpful and even learning rate can be manually set to very small number (like 0.0005 in the table), minimum diverge estimation still improves the result. This is coherent with what we know from the standard i-vector practice.

3.3.2. Comparison between linearity and non-linearity

For non-linearity parametric rectified linear unit[23] is adopted as function g in Equation (4):

$$g_P(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \alpha x, & \text{otherwise} \end{cases} \tag{10}$$

where α is a parameter learned from data and fixed for all dimensions. During training we start α from 1, which correspond to linear case.

Two configurations are tested in our experiments:

$$\begin{aligned} G_1(\mathbf{w}) &= \mathbf{M} + g_p(\mathbf{T}_\theta \mathbf{w}) \\ G_2(\mathbf{w}) &= \mathbf{M} + g_p^1(\mathbf{T_1} g_p^2(\mathbf{T_2}\mathbf{w})) \end{aligned} \tag{11}$$

Method	EER	DCF10^{-2}	DCF10^{-3}
1 Layer PReLU	13.20	0.733	0.862
2 Layer PReLU	14.17	0.755	0.878
Linear	**13.18**	**0.722**	**0.858**

Table 4: *Comparison between linearity and non-linearity. Minimum diverge estimation is adopted in all systems*

Method	EER	DCF10^{-2}	DCF10^{-3}
MLE	14.05	0.748	0.878
MLE+MDE	13.18	**0.722**	**0.858**
MAP	13.44	0.745	0.872
MAP+MDE	**13.11**	0.744	0.876

Table 5: *Comparison between MAP and MLE estimation with/without minimum diverge estimation(MDE)*

where $\mathbf{T_2}$ project $\mathbf{w}$ into an intermediate hidden space with 1024 dimension. g_p^1 and g_p^2 has different parameters.

Results in Table 4 reveals that non-linearity doesn't give improvement. G_1 and G_2 is named as 1 Layer PReLU and 2 Layer PReLU, respectively. We find out that α for g_p^2 is close to 0 which makes intermediate representation extremely sparse. Result suggests linear assumption is good enough for mean shift compared to preliminary non-linear assumption. For non-linear case more further research is still needed.

3.3.3. Comparison between maximum likelihood estimation estimation (MLE) and maximum a posteriori (MAP) estimation

The only difference between MLE and MAP estimation is the weight of prior in Equation (8). Table 5 reports results of MLE and MAP. From the table it is obvious that MAP estimation has similar performance as ML estimation. However, with the introduction of prior term $p(\mathbf{w} = \boldsymbol{\mu})$, $\mathbf{w}$ is closer to standard normal distribution, which reduces the relative improvement of minimum diverge estimation. We further plot distribution of randomly selected dimension in training for both MLE and MAP and plot in Figure 2. From figure $\mathbf{w}$ tends to have similar distribution as standard normal in MAP, compared to MLE. Overall, maximum likelihood estimation gives better decision cost function (DCF) while maximum a posteriori estimation lead to better equal error rate (ERR).

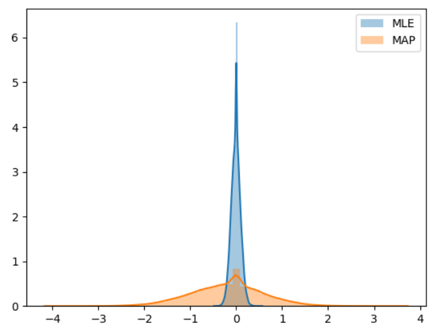

Figure 2: *Distribution of random selected dimension of training non-linear i-vectors. Clearly MAP training lead to distribution more close to standard normal distribution*

4. Conclusions

In this paper we presented a novel stochastic based non-linear *i-vectors* system. The proposed model follows the UBM approach, the same as i-vector. However, it generalizes i-vectors allowing to use a non-linear function to map the latent factor (i-vector) to the utterance dependent GMM super-vector mean. In this kind of model estimating the latent factor or the model parameters is intractable. However, we can resort to a similar approach used in variational autoencoders [18], where model parameters are obtained by stochastic back-propagation and latent factor are computed by a deep neural network. In our framework, we proposed to use stochastic back-propagation to compute both latent factors and model parameters. In the training phase, model parameters and latent factors were estimated iteratively. Meanwhile in evaluation phase–when obtaining enrollment and test i-vectors–, the model parameters were fixed.

We experimented with this framework using linear and non-linear decoders to compute the GMM means. Our experiments on SRE16 show that our framework with both MLE and MAP training lead to slightly better result than conventional *i-vectors*. However non-linear assumption didn't really improve the result. Further work, includes extending non-linearity by allowing different non-linearity in each dimension, and adapting different parameters simultaneously.

5. References

[1] D. A. Reynolds, "Speaker identification and verification using gaussian mixture speaker models," *Speech communication*, vol. 17, no. 1, pp. 91–108, 1995.

[2] P. Kenny, "Joint factor analysis of speaker and session variability: Theory and algorithms," *CRIM, Montreal,(Report) CRIM-06/08-13*, vol. 14, pp. 28–29, 2005.

[3] 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, no. 4, pp. 1435–1447, 2007.

[4] 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. 4, pp. 788–798, 2011.

[5] S. Ioffe, "Probabilistic linear discriminant analysis," in *European Conference on Computer Vision*. Springer, 2006, pp. 531–542.

[6] P. Kenny, "Bayesian speaker verification with heavy-tailed priors." in *Proc. Odyssey*, 2010, p. 14.

[7] D. Garcia-Romero and C. Y. Espy-Wilson, "Analysis of i-vector length normalization in speaker recognition systems," in *Proc. Interspeech*, 2011, pp. 249–252.

[8] Y. Lei, N. Scheffer, L. Ferrer, and M. McLaren, "A novel scheme for speaker recognition using a phonetically-aware deep neural network," in *Proc. ICASSP*. IEEE, 2014, pp. 1695–1699.

[9] P. Kenny, V. Gupta, T. Stafylakis, P. Ouellet, and J. Alam, "Deep neural networks for extracting baum-welch statistics for speaker recognition," in *Proc. Odyssey*, 2014, pp. 293–298.

[10] E. Variani, X. Lei, E. McDermott, I. L. Moreno, and J. Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *Proc. ICASSP*. IEEE, 2014, pp. 4052–4056.

[11] M. McLaren, Y. Lei, and L. Ferrer, "Advances in deep neural network approaches to speaker recognition," in *Proc. ICASSP*. IEEE, 2015, pp. 4814–4818.

[12] N. Chen, Y. Qian, and K. Yu, "Multi-task learning for text-dependent speaker verication," in *Proc. InterSpeech*, 2015, pp. 185–189.

[13] G. Heigold, I. Moreno, S. Bengio, and N. Shazeer, “End-to-end text-dependent speaker verification,” in *Proc. ICASSP*. IEEE, 2016, pp. 5115–5119.

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

[15] F. Chowdhury, Q. Wang, I. L. Moreno, and L. Wan, “Attention-based models for text-dependent speaker verification,” *arXiv preprint arXiv:1710.10470*, 2017.

[16] D. Snyder, D. Garcia-Romero, D. Povey, and S. Khudanpur, “Deep neural network embeddings for text-independent speaker verification,” *Proc. Interspeech*, pp. 999–1003, 2017.

[17] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[18] D. P. Kingma and M. Welling, “Auto-Encoding Variational Bayes,” in *Proceedings of the International Conference of Learning Representations, ICLR 2014*, Banff, Alberta, Canada, apr 2014. [Online]. Available: http://arxiv.org/abs/1312.6114

[19] P. A. Torres-Carrasquillo, F. Richardson, S. Nercessian, D. Sturim, W. Campbell, Y. Gwon, S. Vattam, R. Dehak, H. Mallidi, P. S. Nidadavolu, R. Li, R. R. Pappagari, N. Chen, N. Dehak, and R. Zazo, “The mit lincoln laboratory 2016 speaker recognition system,” in *NIST Speaker Recognition Evaluation 2016*, San Diego, California, Dec. 2016.

[20] P. A. Torres-Carrasquillo, F. Richardson, S. Nercessian, D. Sturim, W. Campbell, Y. Gwon, S. Vattam, N. Dehak, H. Mallidi, P. S. Nidadavolu *et al.*, “The mit-ll, jhu and lrde nist 2016 speaker recognition evaluation system,” *Proc. Interspeech 2017*, pp. 1333–1337, 2017.

[21] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, “The kaldi speech recognition toolkit,” in *IEEE 2011 workshop on automatic speech recognition and understanding*, no. EPFL-CONF-192584. IEEE Signal Processing Society, 2011.

[22] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[23] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.