LANGUAGE RECOGNITION USING DEEP NEURAL NETWORKS WITH VERY LIMITED TRAINING DATA

Shivesh Ranjan, Chengzhu Yu, Chunlei Zhang, Finnian Kelly, John H. L. Hansen

Center for Robust Speech Systems (CRSS)
The University of Texas at Dallas, Richardson, TX, USA
$\{Shivesh.Ranjan, John.Hansen\}$@utdallas.edu

ABSTRACT

This study proposes a novel deep neural network (DNN) based approach to language identification (LID) for the NIST 2015 Language Recognition (LRE) i-Vector Machine Learning Challenge. State-of-the-art DNN based LID systems utilize large amounts of labeled training data. The 2015 LRE i-Vector Machine Learning Challenge limits the access to only ready-to-use i-Vectors for LID system training and testing. This poses unique challenges in designing DNN based LID systems, since optimized front-ends and network architectures can no longer be used. We propose to use the training i-Vectors to train an initial DNN for LID. Next, we present a novel strategy to use this initial DNN to estimate out-of-set language labels from the development data. The final DNN for LID is trained using the original training data, and the estimated out-of-set language data. We show that augmenting the training set with out-of-set labels leads to significant improvement in the LID performance. Our approach obtains very competitive costs (defined by NIST) of 26.56, and 25.98 respectively, on the progress and evaluation subsets of the challenge. Since the amount of training data is very limited (300 i-Vectors per language), this study outlines a successful recipe for DNN based LID using very limited resources.

Index Terms— language identification, i-Vector, deep neural network, limited resources

1. INTRODUCTION

The task of language identification (LID) involves automatically identifying the language in which a given speech utterance was spoken. LID systems are used in a multitude of applications: multilingual language translation, emergency or consumer call routing, surveillance and security applications [1, 2]. Recently, LID has received considerable attention, primarily due to the several NIST Language Recognition Evaluations (LRE), and also in part due to programs such as DARPA Robust Automatic Transcription of Speech (RATS).

This project was funded by AFRL under contract FA8750-12-1-0188 and partially by the University of Texas at Dallas from the Distinguished University Chair in Telecommunications Engineering held by J.H.L. Hansen.

Several approaches to LID have been developed employing Gaussian Mixture Model (GMM) based techniques [3], and Support Vector Machine based approaches [4]. Recently, i-Vector based techniques have become state-of-the-art in LID, closely following similar developments in the Speaker Identification (SID) [5, 6, 7]. More recently, Deep Neural Network (DNN) and Convolutional Neural Network (CNN) based LID approaches are becoming increasingly popular, and have been reported to offer comparable, and in many cases superior performance compared to i-Vector based LID techniques formulated using a Gaussian Mixture Model Universal Background Model (GMM-UBM) based framework [8, 9, 10].

In [11], an i-Vector based LID formulation using bottleneck features extracted from a neural network was shown to outperform a state-of-the-art i-Vector based LID system using Shifted Delta Cepstra (SDC) features. In [8], the GMM-UBM was replaced by a CNN originally trained for Automatic Speech Recognition (ASR), to extract posteriors for an i-Vector based LID formulation. This offered significant improvements compared to a GMM-UBM i-Vector based LID system on the DARPA RATS LID task. A more direct approach to LID using DNNs was proposed in [10]. The approach involved training a DNN to output the language classes, and an additional class for the out-of-set languages. The DNN based LID approach was shown to outperform an i-Vector based LID system when a large amount of labeled training data was available.

The NIST 2015 Language Recognition (LRE) i-Vector Machine Learning (ML) Challenge offers a new paradigm for the development of LID techniques, by limiting the access to only ready-to-use i-Vectors for LID system training, development, and testing [12]. Additionally, the amount of labeled training data is very limited (300 i-Vectors per language). This offers unique challenges in designing a CNN/DNN based LID strategy, since these approaches utilize large amounts of labeled training data: either to train a CNN/DNN for computing the posteriors in a CNN/DNN based i-Vector LID system, or to train a DNN directly for the output language classes. Moreover, these CNN/DNN based

LID techniques use optimal acoustic features such as filter-bank outputs or PLP features [8, 10] which is not feasible in the NIST LRE i-Vector ML Challenge paradigm.

This study proposes a novel approach to LID using a DNN. We propose to train an initial DNN for LID using i-Vectors. Since the test-set contains out-of-set languages with no corresponding labeled training data, we also present a novel out-of-set estimation strategy. The final DNN is trained using the original training data, and the out-of-set language labels estimated from the development data using the initial DNN. The proposed 2-step DNN training approach is shown to offer very competitive costs (defined and evaluated by NIST) on the progress and evaluation sets of the NIST 2015 LRE i-Vector ML challenge, and significantly outperforms a baseline LID system provided by NIST.

2. LANGUAGE IDENTIFICATION (LID) USING I-VECTORS

An i-Vector based LID approach was first introduced in [5, 7]. In the i-Vector paradigm, a language-specific GMM mean supervector M can be represented in terms of the language and channel independent supervector m, a low rank *total variability matrix* T, a vector w, and a residual noise term ϵ as,

$$M = m + Tw + \epsilon. \tag{1}$$

In (1), w is a random vector with a standard normal distribution $N(0, I)$, and ϵ is a residual noise term $N(0, \Sigma)$. The T matrix is learned using large amounts of training data. In the LID framework, utterance-labels correspond to the language of the corresponding utterances. A comprehensive treatment of the i-Vector extraction procedure is presented in [13].

Once the i-Vectors corresponding to the training and test-sets are extracted, several approaches such as a Support Vector Machine (SVM) back-end, Gaussian back-end, or Gaussianized Cosine Distance Scoring (GCDS) can be used to determine the language of the test utterances [5, 14, 15].

3. THE NIST 2015 LRE I-VECTOR MACHINE LEARNING CHALLENGE

The NIST 2015 LRE i-Vector Machine Learning Challenge is aimed at developing new LID techniques employing i-Vectors for conversational/narrow-band broadcast speech [12]. The challenge has 3 distinct data-sets: a training set with 300 i-Vectors per language corresponding to each of the 50 in-set target languages, a test-set, and a development set. The speech utterances corresponding to the training-set i-Vectors were chosen so that their durations exhibit a log-normal distribution with a mean duration of 35.15s. The development and test-set were unlabeled, and also contained i-Vectors corresponding to out-of-set languages. Table 1 shows the number of i-Vectors for the data-sets of the challenge.

Data Set	No. of i-Vectors
Training	15000
Development	6431
Test	6500

Table 1. *Composition of the NIST 2015 LRE i-Vector Machine Learning Challenge data-sets.*

The primary task of the challenge is to identify the corresponding language of a test i-Vector, or to assign it as an "out_of_set" (a single label corresponding to the out-of-set languages), if the i-Vector is deemed not to correspond to any of the 50 in-set languages. The test-set was divided randomly into *progress subset* (30% of the test-set) and *evaluation subset* (70% of the test-set). The challenge rules did not allow using the outputs corresponding to other test i-Vectors to be used in any way in evaluating the output of a given test i-Vector[12]. The performance was assessed using the following cost function defined by NIST,

$$Cost = \frac{(1 - P_{OOS})}{n} * \sum_{k}^{n} P_{error}(k) + P_{OOS} * P_{error}(OOS). \tag{2}$$

In (2), $P_{error}(k) = (\frac{no.\ of\ errors\ for\ class\ k}{no.\ of\ trials\ for\ class\ k})$, $n = 50$, and $P_{OOS} = 0.23$. The cost for *progress subset* and *evaluation subset* were evaluated by NIST.

4. DEEP NEURAL NETWORK (DNN) FOR LID USING I-VECTORS

In [8], large amounts of labeled training data were used to initially train an ASR system, which was then used to generate the senone alignments to train a CNN. In this framework, the CNN/DNN replaced a GMM-UBM to compute the posteriors needed for i-Vector extraction, and subsequent steps of i-Vectors based LID essentially remained unchanged. The front-end for CNN/DNN training utilized filter-bank outputs. A more direct approach to LID using DNNs was outlined in [10], where the output layer nodes correspond to the in-set languages along with a single node for out-of-set languages. This approach also utilized large amounts of labeled training data and employed PLP features.

Our study proposes to train a DNN for LID using i-Vectors unlike existing CNN/DNN based LID techniques. To account for the out-of-set data present in the test-set, we adopt a novel 2-step DNN training strategy, where the initial DNN is trained using only in-set labeled training data. The initial DNN is then used to estimate out-of-set labels from the development data. Next, we train a second DNN for LID with both in-set and estimated out-of-set labels. Moreover, since the amount of training data is very limited, we also investigate and comment on the efficacy of some popular techniques to overcome the issue of limited training data. We have used the PDNN toolkit for the experiments reported in this study [16].

The following sub-sections describe details of the proposed DNN based approach for LID using i-Vectors.

4.1. DNN Training for in-set Languages using i-Vectors

We use a fully connected feed-forward neural network for LID in the experiments reported here. The hidden-layer units use a sigmoid activation function. The output layer is a softmax layer with output nodes corresponding to the in-set languages. Let the target classes be represented as Y, W and b be the weight matrix and bias vector respectively. The output at the ith node of the output layer, corresponding to the input vector x can be expressed as,

$$\begin{aligned} P(Y = i|x, W, b) &= softmax_i(Wx + b) \\ &= \frac{e^{W_i x + b_i}}{\sum_j e^{W_j x + b_j}}. \end{aligned} \tag{3}$$

Next, the predicted language y_{pred} is evaluated as,

$$y_{pred} = argmax_i P(Y = i|x, W, b). \tag{4}$$

The highest score corresponding to the label evaluated in (4) can be obtained using

$$y_{max} = max_i P(Y = i|x, W, b). \tag{5}$$

Since the amount of labeled training data is very limited, we explore several techniques to examine if they offer any improvement in the performance of our DNN based LID system. Of particular interest are techniques of *dropout*, and unsupervised generative *pre-training*, which have been reported to be very effective when training DNNs with limited amounts of data [17, 18].

In the dropout technique, certain units of the hidden layers together with their connections are dropped randomly. This, in turn, minimizes the overfitting in the DNN by reducing the co-adaptation of the network parameters [17]. Unsupervised generative pre-training allows the DNN to use more information from the training data than contained within the labels alone [18]. It has been reported to prevent overfitting by introducing regularization [19], and has been widely used in DNN based ASR techniques [18, 20]. Additionally, L2-norm regularization is also applied since it reduces overfitting by preventing the network weight parameters from assuming very large values.

4.2. Estimating out-of-set Labels for Training from Development Data

The DNN trained on the in-set languages is used to estimate labels corresponding to out-of-set languages from the development data. Specifically, corresponding values of y_{max} are computed using (5) for the i-Vectors of the development set. Next, all i-Vectors with the corresponding scores $y_{max} < \theta$ for some suitable threshold θ (computed using the development set) are assigned the label "out_of_set" (label corresponding to out-of-set data).

4.3. DNN Training with in-set and out-of-set Labels

In the second stage, a new DNN is trained using the i-Vectors for the in-set languages and the i-Vectors corresponding to the out-of-set languages estimated from the development data. Thus, this DNN has an extra node in the output layer compared to the initial DNN to account for the out-of-set languages. The same strategies as mentioned previously in Sec 4.1 are applied to make optimal use of the limited training data. The language labels for the test i-Vectors were assigned using (4). Fig. 1 presents an overview of the proposed DNN based LID approach.

Since the amount of total training data available increased after including the labels for out-of-set languages, we also explored varying the architecture of the DNN compared to what was used for the initial DNN mentioned in Sec. 4.1. Specifically, we explored using more units in the hidden layers as well as deeper networks.

5. EXPERIMENTS, RESULTS AND DISCUSSIONS

An initial DNN for LID was trained using the i-Vectors of the training set. We employed the mini-batch Stochastic Gradient Descent (SGD) algorithm with a mini-batch size of 256, and backpropagation to train the DNN [21]. The DNN had 2 hidden layers with 1024 units each. The input layer had 400 nodes corresponding to the 400-dimensional i-Vectors provided by NIST for the challenge. A dropout factor of 0.2 was used for each of the 2-hidden layers of the DNN. The output layer had 50 nodes corresponding to all the in-set languages. For each of the in-set languages, 10% of the labeled training data (30 i-Vectors per language) was randomly set aside as a *held-out* set to monitor DNN training.

5.1. Effect of out-of-set Detection on the Cost

From (2), it is clear that detecting "out_of_set" labels correctly is critical to obtaining a competitive cost function value on the test-set. In Table 2, the results for two different sets of output-labels are shown for the progress subset. These were

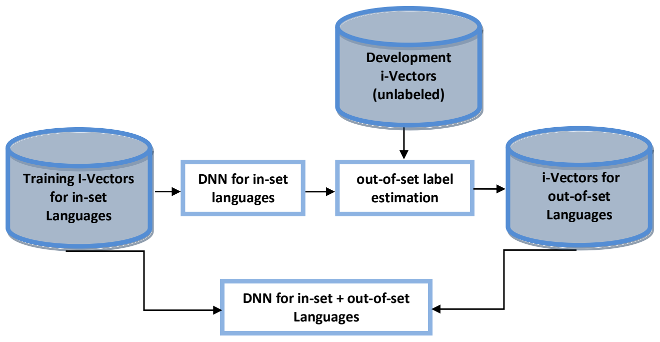
Development i-Vectors (unlabeled)
Training I-Vectors for in-set Languages
DNN for in-set languages
out-of-set label estimation
i-Vectors for out-of-set Languages
DNN for in-set + out-of-set Languages

Fig. 1. *2-step DNN training for LID using i-Vectors.*

obtained using the same initial DNN (DNN1) trained with the in-set languages. The output-labels for "No out_of_set" were obtained using (4) by using the estimated in-set output-labels for the test i-Vectors. Next, using (5), y_{max} was estimated for each i-Vector of the test-set. To obtain the output-labels for "With out_of_set", any y_{pred} with the corresponding $y_{max} < \delta$, for some suitable threshold δ estimated using the development data, was assigned the output-label "out_of_set".

LID System	Output-labels	Cost (*progress subset*)
DNN1	No out_of_set	37.38
DNN1	With out_of_set	32.71

Table 2. *Cost obtained using the initial DNN (DNN1) trained on the in-set languages for output-labels without (No out_of_set), and with the out-of-set (With out_of_set) labels.*

As can be observed from results in Table 2, detecting out-of-set languages in the output offers a big improvement by lowering the cost by 4.67% absolute (12.49% relative).

5.2. LID with DNN trained using in-set and Estimated out-of-set Labels

Table 3 shows results on the *progress subset* using the final DNNs trained on both the in-set and estimated out-of-set labels compared against a Cosine Distance Scoring (CDS) baseline system provided by NIST [12]. Since the estimated labels (1307) for out-of-set languages added approximately 10% more data to the original training-set (13500 i-Vectors for in-set training set), we investigated using more units per hidden layer for the DNN. To this end, Table 3 shows two DNNs: DNN2_1024 with 2 hidden layers each with 1024 units, and DNN2_2048 with 2 hidden layers each with 2048 units. A dropout factor of 0.5 was used for each hidden layer of the 2 DNNs.

LID System	Cost (*progress subset*)
DNN2_1024	26.82
DNN2_2048	**26.56**
CDS (baseline)	39.59

Table 3. *Cost obtained using the DNNs with 1024 (DNN2_1024), and 2048 (DNN2_2048) units per hidden-layer trained using the augmented (in set + estimated out-of-set) training set compared against a CDS baseline system.*

Comparing results of Table 2 and Table 3, including the estimated out-of-set labels in DNN training offered a significant reduction in cost by 5.89% absolute (18% relative), when the cost obtained using DNN2_1024 (26.82) (Table 3) is compared against the "With out_of_set" DNN1 from Table 2. Increasing the units in the hidden layers offered a marginal reduction in cost as evident by the results for DNN2_2048. Both DNN2_1024 and DNN2_2048 are significantly better than the CDS baseline system by over 32% (relative).

DNN2_2048 obtained a cost of **25.98** on the *evaluation subset* of the NIST 2015 LRE i-Vector ML challenge. The results obtained using the proposed DNN based LID approach are comparable to SVM based LID techniques, and offer further improvements in system fusion of the two approaches [22]. We also explored using more than 2 hidden layers for training the DNNs. Adding more layers to the DNN caused degradation in LID performance since the limited training data was insufficient to estimate the new additional parameters.

5.3. Investigating the Efficacy of *Dropout* and Generative *Pre-training* for DNN Training

We investigated the use of dropout and unsupervised generative pre-training for DNN training with limited resources. It was observed that LID performance improved with progressively higher values of the dropout factor from 0.1 to 0.5, after which it started to degrade. A dropout factor of 0.5 for each of the 2-hidden layers achieved the DNN results shown in Table 3.

When DNN2_1024 (trained on in-set and estimated out-of-set labels) was retrained after applying unsupervised generative pre-training (using the unlabeled development set), the cost on the progress subset degraded from 26.82 to 28.46. Evidently, pre-training did not offer any improvement to the proposed DNN based approach for LID. Unlike optimal acoustic features like filter-bank outputs, i-Vectors offer a more compact representation of a speech utterance. We hypothesize that this causes i-Vectors to lose much of the additional information compared to acoustic features such as filter-bank outputs. Since unsupervised generative pre-training works by utilizing the additional information contained within the features [18], it probably fails to access such information when i-Vectors are used. The limited amount of available development data could be another reason why unsupervised generative pre-training failed to offer any improvement.

6. CONCLUSIONS

This study presented a novel approach to LID using very limited training data. To explicitly detect the out-of-set languages, we proposed a novel 2-step DNN training strategy, in which a DNN for LID trained using the in-set labeled training data was used to estimate out-of-set labels from an unlabeled development set. The training set augmented with the out-of-set labels was then used to train a second DNN for LID that could also detect an out-of-set language in addition to the in-set languages. This was shown to offer significantly better LID performance than a DNN utilizing only the in-set labeled data for training. Also, the proposed approach significantly outperformed a CDS based baseline system, and obtained very competitive costs on the progress and evaluation subsets of the NIST 2015 LRE i-Vector Machine Learning Challenge. This study has therefore outlined a successful recipe for DNN based LID using very limited resources.

7. REFERENCES

[1] E. Ambikairajah, H. Li, L. Wang, B. Yin, and V. Sethu, "Language identification: A tutorial," *IEEE Circuits and Systems Magazine*, vol. 11, no. 2, pp. 82–108, 2011.

[2] Y. K. Muthusamy, E. Barnard, and R. Cole, "Reviewing automatic language identification," *Signal Processing Magazine, IEEE*, vol. 11, no. 4, pp. 33–41, 1994.

[3] L. Burget, P. Matejka, and J. Cernocky, "Discriminative training techniques for acoustic language identification," *IEEE ICASSP-2006*, vol. 1, pp. 209–212, 2006.

[4] W. M. Campbell, J. P. Campbell, D. A. Reynolds, E. Singer, and P. A. Torres-Carrasquillo, "Support vector machines for speaker and language recognition," *Computer Speech Language*, vol. 20, no. 2-3, pp. 210–229, 2006.

[5] N. Dehak, P. A. Torres-Carrasquillo, D. A. Reynolds, and R. Dehak, "Language recognition via i-vectors and dimensionality reduction," *ISCA INTERSPEECH*, pp. 857–860, 2011.

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

[7] D. Martinez, O. Plchot, L. Burget, O. Glembek, and P. Matejka, "Language recognition in ivectors space," *ISCA Interspeech*, pp. 861–864, 2011.

[8] Y. Lei, L. Ferrer, A. Lawson, M. McLaren, and N. Scheffer, "Application of convolutional neural networks to language identification in noisy conditions," *Proc. Odyssey-14, Joensuu, Finland*, 2014.

[9] L. Ferrer, Y. Lei, M. McLaren, and N. Scheffer, "Spoken language recognition based on senone posteriors," *ISCA INTERSPEECH*, pp. 2150–2154, 2014.

[10] I. Lopez-Moreno, J. Gonzalez-Dominguez, O. Plchot, D. Martinez, J. Gonzalez-Rodriguez, and P. Moreno, "Automatic language identification using deep neural networks," *IEEE ICASSP-2014*, pp. 5337–5341, 2014.

[11] Y. Song, B. Jiang, Y. Bao, S. Wei, and L. R. Dai, "I-vector representation based on bottleneck features for language identification," *Electronics Letters*, vol. 49, no. 24, pp. 1569–1570, 2013.

[12] NIST, "The 2015 Language Recognition i-Vector Machine Learning Challenge," *Evaluation plan*, 2015.

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

[14] M. Penagarikano, A. Varona, M. Diez, L. J. Rodriguez-Fuentes, and G. Bordel, "Study of different backends in a state-of-the-art language recognition system," *ISCA INTERSPEECH*, 2012.

[15] G. Liu, T. Hasan, H. Boril, and J. H. L. Hansen, "An investigation on back-end for speaker recognition in multi-session enrollment," *IEEE ICASSP-2013*, pp. 7755–7759, 2013.

[16] Y. Miao, "Kaldi+PDNN: Building DNN-based ASR systems with Kaldi and PDNN," *arXiv:1401.6984*, 2014.

[17] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[18] A. R. Mohamed, G. E. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 14–22, 2012.

[19] D. Erhan, Y. Bengio, A. Courville, P. A. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pre-training help deep learning?," *The Journal of Machine Learning Research*, vol. 11, pp. 625–660, 2010.

[20] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[21] T. Zhang, "Solving large scale linear prediction problems using stochastic gradient descent algorithms," *Proc. ICML*, p. 116, 2004.

[22] C. Yu, C. Zhang, S. Ranjan, Q. Zhang, A. Misra, F. Kelly, and J. H. L. Hansen, "UTD-CRSS System for the NIST 2015 Language Recognition i-Vector Machine Learning Challenge," *IEEE ICASSP-2016*, 2016.