INTERSPEECH 2016
September 8–12, 2016, San Francisco, USA

Stacked long-term TDNN for Spoken Language Recognition

Daniel Garcia-Romero and Alan McCree

Human Language Technology Center of Excellence
The Johns Hopkins University, Baltimore, MD 21218, USA
dgromero@jhu.edu, alan.mccree@jhu.edu

Abstract

This paper introduces a stacked architecture that uses a time delay neural network (TDNN) to model long-term patterns for spoken language identification. The first component of the architecture is a feed-forward neural network with a bottleneck layer that is trained to classify context-dependent phone states (senones). The second component is a TDNN that takes the output of the bottleneck, concatenated over a long time span, and produces a posterior probability over the set of languages. The use of a TDNN architecture provides an efficient model to capture discriminative patterns over a wide temporal context. Experimental results are presented using the audio data from the language i-vector challenge (IVC) recently organized by NIST. The proposed system outperforms a state-of-the-art shifted delta cepstra i-vector system and provides complementary information to fuse with the new generation of bottleneck-based i-vector systems that model short-term dependencies.

Index Terms: language recognition, deep neural networks, long-time span

1. Introduction

Modeling very long-term acoustic dependencies (in the order of seconds) for spoken language recognition is a largely unexplored area, mostly due to the difficulty in designing models that can efficiently capture such long range dynamics. In this work we propose a stacked architecture that uses a time delay neural network (TDNN) [1] to accomplish this goal. The first component of the architecture is a feed-forward neural network with a bottleneck (BN) layer that is trained to classify context-dependent phone states (senones). BN features, in combination with an i-vector system, are becoming the next generation language recognition systems due to their outstanding performance [2, 3, 4, 5, 6]. Our architecture takes advantage of the discriminative power of the BN features, but our second component (TDNN) is trained to directly produce frame-by-frame posteriors over the set of languages. In this way, the TDNN is replacing the i-vector classifier in a similar way as in other direct systems [7, 8, 9]. However, the use of BN features and a much wider temporal context differentiates our proposed system from previous work. Out stacked architecture is trained in two stages like other stacked systems [10, 5]. This approach simplifies the training greatly.

The remainder of the paper is organized as follows. Section 2 describes the baseline systems and summarizes the role of the TDNN. Section 3 describes our datasets and metrics. Section 4 presents the procedure to train the systems, and the experimental results. Finally, section 5 provides the conclusions.

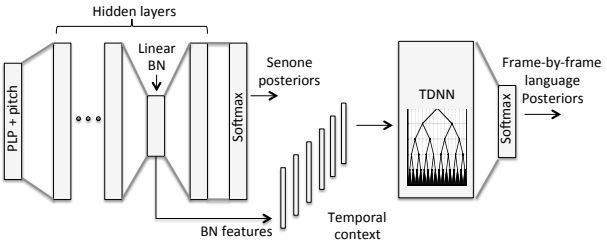

Figure 1: Block diagram of the stacked TDNN architecture.

2. Language Recognition Systems

2.1. Baseline i-vector system

Most state-of-the-art systems for i-vector language recognition use classifiers such as Gaussian models, logistic regression, or cosine scoring, followed by a multiclass back-end which provides significant performance improvement as well as producing calibrated probability outputs. We have recently demonstrated success using only a single step: a Gaussian classifier discriminatively trained using Maximum Mutual Information (MMI) [11]. This system serves as the acoustic baseline for this paper.

2.2. Bottleneck i-vector system

Hand-crafted shifted delta cepstra features (SDC) derived from MFCCs used to be the best feature representation for language recognition. Recently, BN features extracted from a feed-forward neural network that is trained to classify context-dependent phone states (senones) have been shown to greatly outperform SDC features [2, 3, 4, 5]. The intuition for their success is that, since they are trained to discriminate between fine-grained phonetic categories, they must capture phonetically rich information with some invariance to channel and speaker variability. The first block of Figure 1 shows the DNN that is used to compute the BN features. Note, that the BN layer is linear. It is possible to use a non-linear BN, but our experience has been that linear BN layers produce features that are easier to model by the GMM of the i-vector extractor. Once, the BN features are extracted, the BN i-vector system pipeline is identical to our baseline system.

2.3. Stacked TDNN system

As shown in Figure 1, the stacked TDNN system takes advantage of the superior discriminative power of the BN features, and uses a TDNN to directly produce frame-by-frame posteriors over the set of languages. A cut level score can be obtained by averaging the input vectors of the LID softmax layer. Note

http://dx.doi.org/10.21437/Interspeech.2016-1334

that the large temporal context makes the frame-by-frame posterior very redundant and smooth across time. This allows for a large temporal downsampling (i.e. strides much bigger than 1 frame) when producing a cut level score.

The TDNN is capable of efficiently modeling wide temporal contexts due to its hierarchical multi-resolution nature. In particular, the input layer processes a narrow temporal context that keeps expanding as the information flows towards higher layers [1]. The configuration of the hierarchical structure controls how the information is processed. Figure 2 shows a symmetric architecture of a 5 layer TDNN that processes a context of 128 frames. The branches of the tree indicate the temporal indices of the inputs to the layer. That is, the input to layer 5 is the concatenation of the output of layer 4 from time indices -32 and 32 relative to the center frame being processed (time index 0). Continuing down in the hierarchy, we see that layer 4 only proceses 2 inputs during the entire timespan of 128 frames. The first one is constructed by concatenating the output of layer 3 at time indices -48 and -16. The second one is the symmetric counterpart with indices 16 and 48. As we progress further down, we can see that the information being processed corresponds to narrower contexts. Also, the parameters of the weight matrices are tied across the inputs to the layer. That is, the 2 inputs processed by layer 4 were multiplied by the same weights. Parameter tying and the sparse temporal sampling of the hierarchical structure are responsible for the efficiency of the TDNN.

3. NIST language i-vector challenge

3.1. Datasets

In 2015 NIST coordinated the first language recognition evaluation that distributed i-vectors to participants [12]. The speech data used to produce the i-vectors was collected over the telephone channel (telephone conversations and narrowband broadcasts) and comprised 65 languages. The official task for the IVC was open-set language identification. The task was defined using 3 datasets: train, dev, and test. The train set provided labeled data for 50 languages. The dev set had data from all 65 languages, but the labels were not provided. Finally the test set had 100 cuts from each of the 65 languages and was used to evaluate the systems.

For this work, we are not interested in the open-set component of the evaluation, but still want to take advantage of the predefined test set with the largest number of languages available to us. Therefore, we are going to merge the train and dev sets and use their labels for the 65 languages. Moreover, our colleges from Lincoln Laboratories (who organized and provided the audio for the IVC) shared an additional set that had more labeled data for some of the languages but was highly unbalanced. We refer to this set as the background (bkg) set. For our system we will use the bkg+dev+train set with all the labels. Note that regardless of all the data reorganization, the test set is kept the same as in the official IVC.

3.2. Metrics

Since the test set comprises 100 cuts per language, the experimental setup can be used to produce 6500 identification trials, or $6500 \times 65 = 422500$ detection trials. To keep the connection with the IVC we evaluate hard identification decisions in terms of probability of error Pe. Also, as is customary in regular NIST language recognition evaluations (LRE) that evaluate detection, we use the standard C_{avg} metric [13].

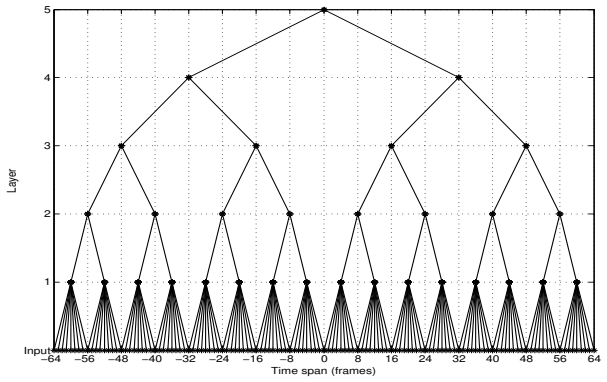

Figure 2: Temporal subsampling of a TDNN with 5 hidden layers and a total input context of 128 frames.

4. Experiments

4.1. System setups

4.1.1. Baseline

The baseline system for these experiments is the acoustic i-vector system presented in [11]. Feature processing uses 24 Mel Frequency Cepstral Coefficients (MFCC) from 0-4 kHz, windows of 25 ms length with 10 ms shift, vocal tract length normalization (VTLN) trained with four iterations of speaker adaptive training, RASTA filtering, conversion to MFCCs, shifted delta cepstra (SDC) coefficients with 7-1-3-7 configuration, static cepstra appended to produce a 56-dimensional feature vector, gating with a GMM-based speech activity detector, and feature vector mean and variance normalization with a 3 second sliding window. The resulting feature sequence is then aligned to a 2048 mixture GMM trained on all the available (bkg+dev+train) cuts of duration longer than 30 seconds, and a 600-dimensional i-vector is estimated using an i-vector extractor trained on the same set. Whitening and length-normalization [14] are applied, followed by diagonalized LDA dimension reduction to 64 dimensions. A multiclass Gaussian classifier is used for language recognition in i-vector space, with the shared within-class covariance and class means first estimated by the sample covariance and means, followed by discriminative refinement of first the covariance scale factor and then the class means using MMI. All the components in the i-vector space pipeline were trained using all the data (bkg+dev+train), except for the refinement step which only used the train subset (as it was found to work better).

4.1.2. Bottleneck

We trained our DNN using the Kaldi speech recognition toolkit (nnet2 framework). The BN DNN is trained to minimize cross-entropy with respect to a set of senone labels. The labels (i.e. frame alignments to senones) are obtained from a standard tied-state triphone GMM-HMM system trained with maximum likelihood on the Fisher English database. The senone set is obtained by clustering the states using a decision tree and the number of total Gaussians was set to 300K. The number of senones after the clustering was 7611. The input features for the GMM-HMM system are 40 dimensional vectors obtained from an LDA+MLLT projection of 7 spliced frames of 13 MFCCs. These features are further processed by an fMLLR transform to perform speaker adaptation.

Our BN architecture uses 6 hidden layers of 2048 sigmoid nonlinearities. An 80 dimensional linear bottleneck is placed between the 5th and 6th sigmoid layers. The input to the DNN are 25 spliced vectors of 13 plp + 3 pitch features [15].

The DNN training algorithm performs back propagation using mini-batch pre-conditioned gradient descent and parameter averaging [16]. Data parallelization is accomplished by training n replicas of the DNN ($n = 4$ for our system) independently on disjoint subsets of data and combining them periodically by averaging their parameters. Once a new updated DNN is obtained, it gets replicated and asynchronous training is performed again. We refer to each one of these stages of "replicate-train-merge" as an iteration. An epoch (i.e. a run over the entire training dataset) consists of a fixed number of these iterations. During an iteration, each replica does back propagation over 300,000 training examples (using mini-batches of size 512). We use an exponential decay schedule for the learning rate and minimize negative cross entropy for a fixed number of epochs. Convergence is monitored on a validation set in terms of cross entropy and frame accuracy.

To obtain the BN features we strip the layers after the linear bottleneck and perform a forward pass on the data. The number of parameters after stripping the layers is reduced from 33 million to 18 million. The resulting features are post-processed using a per cut mean and variance normalization. Then, these BN features are used in the baseline system instead of the SDC features. All the components of the pipeline are identical to the baseline system to compute and classify the i-vectors.

4.1.3. Stacked TDNN

The first component of the stacked TDNN is the BN DNN described above. The second component is a TDNN that uses p-norm non-linearities with $p = 2$ and an input/output dimension pooling ratio of 10 (e.g. 3000/300) [17]. The parameter-tying across time provides an efficient model for long time spans without an explosion of the number of parameters. Figure 1 shows the pattern of the temporal splicing used in this work. Increasing or reducing the number of hidden layers allows the time span to change while keeping the same structure in the layers. The number of parameters depends on the size of the p-norm layers, number of layers, and splicing architecture. In the experiments we explore different configurations of these parameters.

The frame-by-frame nature of this system exacerbates the problem of training data imbalance. In our dataset, the number of training cuts per language is already imbalanced and they have widely variable durations. To alleviate this problem, we used two strategies. First, we truncated cuts longer than 60 seconds. Second, we used a weighted cross-entropy to balance the objective function and reduce the dominance of a subset of languages. The smallest weights were given to English and Vietnamese (0.17). Balancing the training set using these techniques was beneficial in terms of performance and training time.

To generate training examples, we pasted together all the frames from the same cut after removing the silence. This procedure introduces edge effects but greatly simplifies the process. The same approach is used for the test data. As future work we plan to examine if this decision has detrimental impact in terms of performance.

Using a 1 frame shift, the total number of potential training examples from the bkg+dev+train data is approximately 255 million (with a median of 2.5 million frames per language). Note that by example we mean a contiguous block of BN features equal to the time-span of the TDNN (e.g. 512 frames of BN features). Due to the large temporal context, these examples are highly redundant and care must be taken when organizing the data for each iteration. To create the training data, we shuffled the examples so that a model replica would see diverse and minimally redundant data for each iteration (i.e. small temporal overlap between examples). Also, we distributed training examples with small time-shifts across the datasets for each replica to facilitate convergence of the model averaging step (i.e. keeping the replicas from moving too far from each other due to very different training data per iteration).

In this work, we have trained the stacked TDNN system in two stages. The first stage trains the BN DNN as described in the previous section. The second stage treats the output of the BN component as features and only trains the TDNN layers. The TDNN training stage also uses mini-batch pre-conditioned gradient descent and parameter averaging [16]. During an iteration, each replica does back propagation over 300,000 training examples (using mini-batches of size 512). We set an exponential decay schedule for the learning rate and minimize negative cross entropy. Convergence is monitored on a validation set in terms of cross entropy and frame accuracy. However, unlike in the BN case, we do not run for a fixed number of epochs. Instead, we use a counter for the number of unsuccessful iterations where the cross-entropy on the validation set does not improve. Every 5 failed attempts we halve the learning rate and restart the process using the model with the best validation cross-entropy. We allow 5 of these coarse reductions in learning rate before we stop the training. The final network is the one that provides the best validation error during this process.

The validation set plays a significant role in this approach. We designed it to comprise 500 training example for each of the 65 languages. The examples for each language were extracted from 10 randomly selected cuts from the full set of bkg+dev+train data. The selected cuts were not included in the DNN training set. Comparing this learning schedule with the simpler exponential decay, we observed that the final performance was the same and the training time was significantly reduced (from one day to around 6 hours, in both cases using 4 GPUs).

In order to obtain per cut scores on the test data, we average over time the 65 dimensional vectors prior to the softmax component of the TDNN. We treat this average as a vector of language log-likelihoods. For test cuts shorter than the temporal context of the TDNN, we pad the input by repeating the first and last frames evenly. Multi-class calibration using a common scale and a per language bias is trained on the validation data with the Focal toolkit [18]. For the language identification task (measured by probability of error P_e), we pick the language with the largest log-likelihood in the score vector. For the language detection task (measure by C_{avg}), we apply Bayes rule to the calibrated vector of scores and then map it back to a vector of detection log-likelihoods.

4.2. Results

4.2.1. Performance comparison

The top two rows of Table 1 show the performance of the i-vector systems: the baseline system that uses SDC features, and the newer generation of systems that use BN features from a DNN trained for speech recognition. Although, the baseline system represents what until very recently was considered as a strong state-of-the-art system (with VTLN and SDC features), the BN i-vector system greatly outperforms it (approximately

Table 1: Performance results for the systems as well as the number of parameters they use. The * indicates that an oracle (trained on the test data) multi-class calibration has been applied. This serves as an optimistic upper bound on performance.

Systems	Number of parameters (in millions)				TDNN p-norm	Pe (%)	C_{avg} x 100	Pe* (%)	C^*_{avg} x 100
	T mtx	BN	TDNN	Total					
Baseline	69	-	-	69	-	14.7	3.12	14.1	2.84
BN i-vec	98	18	-	116	-	8.9	1.91	8.7	1.73
Stacked TDNNs 512 frames of context	-	18	2	20	1000/100	15.0	2.42	13.2	2.09
	-	18	6	24	2000/200	13.3	2.09	11.7	1.81
	-	18	13	31	3000/300	11.4	2.03	10.1	1.66
	-	18	22	40	4000/400	11.1	1.99	9.6	1.52
	-	18	34	52	5000/500	10.8	1.92	9.6	1.55
	-	18	48	66	6000/600	10.9	1.93	9.3	1.60

40% relative gain in both C_{avg} and P_e). This behavior is consistent with our results (and those of other participants) in the recent NIST LRE15 evaluation [6].

Our proposed stacked TDNN architecture also takes advantage of the BN features, but it is designed to learn discriminative patterns over a wider temporal context (up to 5 seconds in this work). Moreover, the TDNN component is trained to directly produce posterior probabilities over the set of target languages. The bottom part of Table 1 shows the performance of different stacked TDNN systems where the number of parameters is increased by changing the size of the p-norm component. All the networks use 7 hidden layers with the splicing pattern showed in Figure 1, and model a temporal context of 512 frames. The performance of the systems increases with the number of parameters but seems to saturate for a p-norm size of 5000/500. The relative gains of the best TDNN with respect to the baseline are also around 40% in C_{avg}, but slightly smaller for the P_e (27%). These results are obtained with a number of parameters smaller than the baseline system (52 vs 69 million), and almost 50% less than the BN i-vector system. Note, that in terms of C_{avg} the BN i-vector and the stacked TDNN systems perform the same, but the stacked TDNN is worse in terms of P_e. This seems to indicate that the BN i-vector system might not be well calibrated. However, after applying an oracle (trained on the test data) multi-class calibration on the scores using Focal the trends are the same (results shown with an *). This legitimizes the observation that the stacked TDNN is performing very well for the detection task measured by C_{avg}.

4.2.2. Fusion gains

Although the stacked TDNN system performs very well on its own, the difference in training criterion (directly targeting the languages) as well as the temporal context makes it a good candidate to complement the BN i-vector system. To verify this hypothesis, we performed a simple sum fusion of their scores (i.e. summed the scores and divided by 2). The fusion results in a P_e of 7.7 (14% gain) and a C_{avg} of 1.41 (26% gain).

4.2.3. Temporal context analysis

Table 2 shows the performance of stacked TDNNs with p-norm 3000/300 as a function of the temporal context modeled. The time span was changed by keeping the splicing pattern fixed and reducing the number of hidden layers. It is clear that the stacked TDNN system benefits from looking at large temporal contexts. The performance increases as the temporal context increases.

Table 2: Performance of stacked TDNNs with p-norm 3000/300 as a function of the temporal context modeled. The time span was changed by keeping the splicing pattern fixed and reducing the number of hidden layers.

TDNN Context	Hidden layers	Parameters (millions)	Pe (%)	C_{avg} x100
512	7	13	11.4	2.03
256	6	11	13.9	2.23
128	5	9	16.2	2.66
64	4	8	18.5	3.13

It could be possible that this is an artifact due to the way the context was decreased by removing hidden layers. However, modifying the splicing pattern to keep the number of hidden layers constant resulted in worse performance. Also it is the case that the number of parameters of the TDNNs with shorter context is smaller. But Table 1 shows that a TDNN with p-norm 2000/200 and a context of 512 frames used 6 million parameters and still outperforms the shorter context TDNNs that have more parameters. Finally, it seems that increasing the context beyond 512 frames might be a good way to keep improving performance. For this work, we did not explore that option since 20% of the test data is shorter than 10 seconds, and as usual the errors are mostly coming from short test segments.

5. Conclusions

In this paper, we have introduced a stacked architecture that uses a time delay neural network (TDNN) to model long-term patterns for spoken language identification. Experimental results were presented using the audio data from the language i-vector challenge. We evaluated the proposed system in a closed-set language identification task, as well as in a detection task. The proposed system outperforms a baseline SDC i-vector system by 40% in terms of C_{avg} and provides a relative gain of 27% in identification accuracy. Moreover, the stacked TDNN matches the detection performance of a BN i-vector system that models short-term dependencies. A fusion of the stacked TDNN system with the BN i-vector system results in significant gains. This validates the ability of the stacked TDNN system to capture discriminative long-term dependencies that complement the shorter-term patterns captured by the BN i-vector system.

6. References

[1] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *Interspeech*, 2015.

[2] Y. Song, B. Jiang, Y. Bao, S. Wei, and L. Dai, "I-vector representation based on bottleneck features for language identification," in *IEEE Electronic Letters*, 2013.

[3] P. Matejka, L. Zhang, T. Ng, S. H. Mallidi, O. Glembek, J. Ma, and B. Zhang, "Neural network bottleneck features for language identification," in *Odyssey: The Speaker and Language Recognition Workshop*, 2014.

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

[5] R. Fer, P. Matejka, F. Grezl, O. Plchot, and J. Cernocky, "Multilingual bottleneck features for language recognition," in *Interspeech*, 2015.

[6] A. McCree, G. Sell, and D. Garcia-Romero, "Augmented data training of joint acoustic/phonotactic DNN i-vectors for NIST LRE15," in *Odyssey: The Speaker and Language Recognition Workshop*, 2016.

[7] I. Lopez-Moreno, J. Gonzalez-Dominguez, O. Plchot, D. Martinez, J. Gonzalez-Rodriguez, and P. Moreno, "Automatic language identification using deep neural networks," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2014.

[8] 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*, 2014.

[9] J. Gonzalez-Dominguez, I. Lopez-Moreno, P. Moreno, and J. Gonzalez-Rodriguez, "Frame-by-frame language identification in short utterances using deep neural networks," in *Neural Networks*, vol. 64, 2015.

[10] K. Vesely, M. Karafiat, F. Grezl, M. Janda, and E. Egorova, "The language-independent bottleneck features," in *IEEE Spoken Language Technology Workshop (SLT)*, 2012.

[11] A. McCree, "Multiclass discriminative training of i-vector language recognition," in *Proc. Odyssey*, 2014, pp. 166–172.

[12] A. Tong, C. Greenberg, A. Martin, D. Banse, H. Zhao, G. Doddington, D. Garcia-Romero, A. McCree, D. Reynolds, E. Singer, J. Hernandez-Cordero, and L. Mason, "2015 NIST language recognition i-vector machine learning challenge," in *Odyssey: The Speaker and Language Recognition Workshop*, 2016.

[13] "The NIST year 2009 language recognition evaluation plan," http://www.itl.nist.gov/iad/mig/tests/lang/2009/LRE09_EvalPlan_v6.pdf, 2009.

[14] D. Garcia-Romero and C. Espy-Wilson, "Analysis of i-vector length normalization in speaker recognition systems," in *Interspeech*, Florence, Italy, August 2011.

[15] P. Ghahremani, B. BabaAli, D. Povey, K. Riedhammer, J. Trmal, and S. Khudanpur, "A pitch extraction algorithm tuned for automatic speech recognition," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2014.

[16] D. Povey, X. Zhang, and S. Khudanpur, "Parallel training of deep neural networks with natural gradient and parameter averaging," in *International Conference on Learning Representations (ICLR), submitted*, 2015.

[17] X. Zhang, J. Trmal, D. Povey, and S. Khudanpur, "Improving deep neural network acoustic models using generalized maxout networks," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2014.

[18] N. Brummer, "Focal multi-class: Toolkit for evaluation, fusion and calibration of multi-class recognition scores," in *Technical Report*, 2007.