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

Sequence Summarizing Neural Networks for Spoken Language Recognition

Jan Pešán, Lukáš Burget, Jan "Honza" Černocký

Brno University of Technology, Speech@FIT group and IT4I Centre of excellence, Czech Republic
{ipesan,burget,cernocky}@fit.vutbr.cz

Abstract

This paper explores the use of Sequence Summarizing Neural Networks (SSNNs) as a variant of deep neural networks (DNNs) for classifying sequences. In this work, it is applied to the task of spoken language recognition. Unlike other classification tasks in speech processing where the DNN needs to produce a per-frame output, language is considered constant during an utterance. We introduce a summarization component into the DNN structure producing one set of language posteriors per utterance. The training of the DNN is performed by an appropriately modified gradient-descent algorithm. In our initial experiments, the SSNN results are compared to a single state-of-the-art i-vector based baseline system with a similar complexity (i.e. no system fusion, etc.). For some conditions, SSNNs is able to provide performance comparable to the baseline system. Relative improvement up to 30% is obtained with the score level fusion of the baseline and the SSNN systems.
Index Terms: Sequence Summarizing Neural Network, DNN, i-vectors

1. Introduction

Spoken language identification (LID) is an important part of speech data mining. The task is to select the language spoken in an utterance. Typically, it is considered as a closed-set problem, i.e. one language from N possible ones should be chosen. The traditional approaches to LID include acoustic and phonotactic ones, see for example [1].

Nowadays state-of-the-art LID systems make use of i-vectors defined originally for speaker verification [2]. This generative approach is based on a Gaussian Mixture Model (GMM) with means adapted from a Universal Background model (UBM) towards the current utterance in a low-dimensional subspace. The utterance-specific latent variable (an i-vector) is a fixed- and low-dimensional representation of the utterance and actually jointly represent several of its properties (speaker, language, etc.). In i-vector based LID [3], the i-vector is considered as a feature vector to the following classifier (e.g. Multiclass Logistic Regression). Note that the i-vector model assumes independent and identically distributed (i.i.d.) frames, allowing for estimation of the i-vector from sufficient statistics collected from the utterance. It ignores evolution of speech in time.

The recent success of deep neural networks (DNN) in speech recognition has influenced also LID and we have seen DNNs at places of different building blocks of LID systems in recent years. Lei et al. [4] replaces the GMM in an i-vector system with Convolutional Neural Network (CNN) trained in a standard ASR-fashion to estimate posterior probabilities of tied-states (senones). This approach is actually moving the system from acoustics back to phonotactics and the authors report its good performance in noise conditions and complementarity with acoustic i-vectors. Lopez-Moreno et al. [5] train the DNN as a per-frame language classifier and they average obtained posteriors over the utterance to yield the final posteriors. They report 70% relative improvement in *Cavg*[1] using conventional (PLP) features. However, large amount of training data for each language is needed to benefit from this approach. Also, the improvements were seen only for very short test utterances (around 3s). For longer utterances, the techniques fails to provide a competitive performance.

In our opinion, both approaches do not make full use of the DNN capabilities: the first still relies on a different classifier. The DNN is taken from an ASR system and it is not trained for the target task. In the second one, training of the neural network in per-frame manner is sub-optimal, as LID should ultimately produce *one set of class (language) posteriors* for one utterance.

A recurrent neural network (RNN) is a candidate for such a solution: it can be configured to give us the decision only at the end of an utterance. Unfortunately, RNN approach has problem with vanishing gradients due to presence of recurrent connections between layers. Gonzales-Dominguez et al. [7] have experimented with an RNN (more precisely Long Short-Term Memory - LSTM) approach to LID, but had to recur to classification of short chunks in order to randomize the gradients.

Our paper presents the use of a sequence-summarizing neural network (SSNN) for LID. SSNN is a variant of DNN for producing one set of posteriors for the whole utterance. It is a standard feed-forward neural network with summarization layer positioned in the middle of the network. SSNN was first defined by Vesely et al. [8] to obtain adaptation vectors for an ASR task. In their scheme, a separate summarizing DNN provided a constant summary vector, which was fed to the main classification DNN simultaneously with the speech features. Both networks were trained jointly to optimize the final ASR criterion. Zmolikova et al. [9] have successfully used SSNN for data selection in mismatched training-test ASR setups.

For the LID task, we have simplified the SSNN - our structure is shown in Fig 1. As in i-vector approach, the output from the summarization layer –the summary vector – is also a fixed-length representation of an utterance. The summary vector extractor and its classifier is jointly trainined, i.e. it is a fully discriminative end-to-end approach. SSNN is not influenced by re-ordering feature vectors in time, the summary layer weights them all equally. Therefore, on contrary to RNN, it can be compared to i-vector extraction from statistics collected from the utterance, and we expect SSNN to work for the same class of problems as i-vectors.

Note that recent years have seen advances in feature extraction for the LID task. While traditional systems use MFCC, PLP or shifted-delta cepstra (SDC), Fer et al. [10] have found

[1] *Cavg metric was first introduced in NIT LRE 2009. For more details see [6]*

http://dx.doi.org/10.21437/Interspeech.2016-764

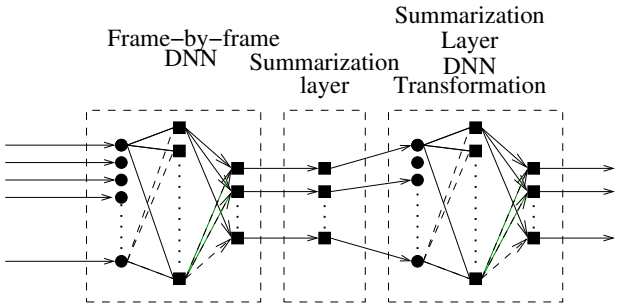

Figure 1: SSNN general structure

Bottleneck Features (BN) [11] imported from an ASR system to perform the best for LID. We have to take into account however, that the BN-DNN for feature extraction must see a huge amount of (possibly multi-lingual) transcribed data to be properly trained. Therefore, for sake of fairness, we compared the SSNN (which sees only the target LID data for its training) with a simple MFCC-SDC/i-vector system.

2. Sequence Summarizing Neural Networks

In LID, given a sequence $\mathbf{X}$ of length of t frames $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_t]$, our task is to find a mapping for each sequence $\mathbf{X}$ to appropriate class from the set of C classes.

A standard neural net layer is defined as

$$\mathbf{O}_n = h_n\left(\mathbf{W}_n\mathbf{O}_{n-1} + \mathbf{b}_n\right), \tag{1}$$

where $\mathbf{O}_n$ is the output matrix from *n-th* layer, $\mathbf{W}_n$ is the weight matrix for this layer, $\mathbf{b}_n$ is the vector of biases, h_n is an element-wise activation function and $\mathbf{O}_{n-1}$ is the output matrix from the previous layer $n-1$. The input to the first layer $\mathbf{O}_0$ is the feature matrix $\mathbf{X}$. A distinguishing element of our SSNN is the summarization pseudo-layer, which does not contain any trainable parameters. It collapses the input sequence into a fixed-length *summary vector*. In this work, the summarization layer simply calculates the mean of the input vector sequence:

$$\mathbf{o}_s = \frac{1}{T}\sum_{t=0}^{T}\mathbf{O}_{s-1}^{(t)} \tag{2}$$

SSNN is defined as a sequence of standard neural net layers (Eq. 1) with one layer replaced with summarization pseudo-layer.

To obtain posterior probability of class $p(c|\mathbf{X})$ we use *softmax* function on the output from the last layer $\mathbf{o}_n$

$$p(c|\mathbf{X}) = \frac{\exp(\mathbf{o}_n(c))}{\sum_{m=1}^{C}\exp(\mathbf{o}_n(m))} \tag{3}$$

2.1. Training

Let us have a loss function

$$L = \log p\left(c_t|\mathbf{X}\right) \tag{4}$$

with $\log p\left(c_t|\mathbf{X}\right)$ being log posterior probability for true class label c_t given one training example – the whole feature matrix $\mathbf{X}$.

Derivatives of (4) with respect to one particular parameter θ from the part *after* summarization are obtained in the same manner, as in the case of propagating one frame through standard feed-forward neural network, so we do not need to discuss it any further.

Derivatives of (4) with respect to one particular parameter from the part *before* summarization θ_{s-1} are obtained as

$$\frac{\partial L}{\partial \theta} = \frac{\partial L}{\partial \mathbf{o}_s}\frac{1}{T}\sum_{t=0}^{T}\frac{\partial \mathbf{O}_{s-1}}{\partial \theta} \tag{5}$$

As mentioned above, RNN suffers from a problem of gradient vanishing/explosion. This is caused primarily by having recursive connections through the network. While doing Backpropagation through time (BPTT) [12], every time-step gradient is propagated recursively through non-linearities which can cause gradient vanishing or explosion depending on initial conditions. From Equation 5 we see, that there are no recurrent functions in SSNN derivatives. This means that we can control the gradient vanishing/explosion problem simply by setting reasonable number of layers.

Unlike per-frame approaches, SSNN has significantly smaller number of weight updates. This is caused by the summarization pseudo-layer, which enforces just one update per utterance. It raises a risk of over-fitting the network, if sufficient amount of training utterances is not available.

3. Experiments

Experiments were carried out on NIST Language Recognition Evaluation 2015 [13] data. The data contains 20 target languages divided into 6 language clusters (Arabic, Chinese, English, French, Slavic, Iberian) and the task is to perform a dialect identification within a language cluster.

3.1. Data

In this paper, we report results only on the primary condition, where data for training is restricted - 394 hours of recordings in total. Originally, only the training set was provided by NIST as it is usual in all evaluations; we further divided it into one smaller training set with 248 hours in 3042 segments and development set with 146 hours in 42295 segments. For training the SSNN classifier, the training was further sub-divided into two subsets with 10% of files used as cross-validation held out set. This sub-division yielded us 2729 utterances for training of SSNN and 313 utterances for cross-validation with the same proportions of languages in both subsets. The development set was used as a held out set to develop the system before evaluation data were released. It was used to report performance, and for estimating calibration parameters used to calibrate evaluation scores. Distribution of utterance lengths in the development set tries to simulate presumed eval set utterance durations. The details of the dataset division can be found in [14]. The evaluation set delivered lately contains 164334 utterances.

3.2. Baseline system

As the reference features, we use popular SDC features [15] with usual configuration 7-1-3-7, concatenated with 7 MFCC coefficients (including C0). The frame rate is 10 ms. Cepstral mean and variance normalization (CMVN) and RASTA filtering [16] are applied before SDC.

For UBM training, we used subset of training set which consisted of 145 hours of speech, with duration per language limited to 15 hours. The UBM has 2048 Gaussians components with full-covariance matrices. The i-vector extractor was

trained on the full training set with 248 hours of speech. The resulting dimensionality of i-vectors is 600. We used conventional setup with a Gaussian Linear Classifier used for classification of i-vectors. The output scores were calibrated using multi-class logistic regression.

3.3. SSNN System

For Sequence Summarizing setup, we use 40 log Mel filterbank features, with frame context of ± 15 frames. Stacked frames are projected into 16 Hamming-weighted DCT temporal basis, which gives us the input to the neural net - vectors of $31 \times 16 = 496$ dimensions. Utterance based mean and variance normalization is applied on the input to the SSNN. The complete LID SSNN structure is shown in Fig. 2.

3.3.1. RMSPropAvg optimizer

To optimize our error function we found *RMSProp* optimizer [17] to be very effective. This optimization method works the best when minibatches are randomized properly. Normally, frames from the training set are assembled into one huge matrix, randomized and cut into minibatches together with according labels. In our case, one training example is a sequence of variable length, which makes creating of minibatches in standard way difficult. Moreover, randomization of frames within one sequence does not make sense, as the resulting gradient is always the sum of gradients from all frames, as shown in Eq. 5.

Masking approach can be used for training with variable length sequences. It utilizes two input matrices for training: a mask and a feature matrix. In this architecture, maximal allowed sequence length is pre-defined. Then, the mask matrix and the feature matrix are allocated in such a way, that the mask determines which frames in feature matrix are *valid* for the training. However, this is memory inefficient and particularly slow on Graphical Processing Unit (GPU).

Because of this, we are forwarding just one utterance at a time and accumulate gradients on every layer. After sufficient number of utterances (i.e. the number of utterances we want to include into one minibatch) we perform an update of weights. This is equivalent to masking, but without the necessity of extracting the features vectors from a sparse masked matrix. It proves to be faster and more efficient than masking.

3.3.2. Training

We used two approaches for training the systems. The first one, *"percluster"*, exploits the way how LRE15 evaluation is scored. In LRE15, average of target language clusters *Cavg* is computed. Therefore, six language cluster-dependent systems were trained and outputs from individual systems were concatenated to obtain the final scores. In the second approach, the training is done in usual way: one system is trained to discriminate between all languages from all language clusters.

For *percluster* systems, we saw a strong tendency to overfitting. This happens for two main reasons. First, *percluster* systems see only a small fraction of the training set belonging to the corresponding language cluster. Second, we have just one update per utterance caused by summarization pseudo-layer. Even after heavy regularization, we were not able to force the system to generalize well.

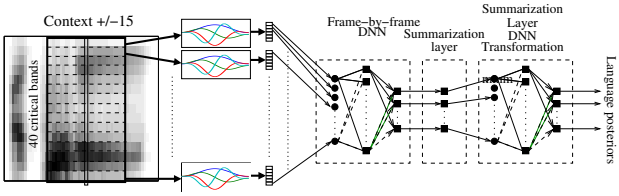

Figure 2: Structure of SSNN for LID

Table 1: Individual systems' and fusion results

System	Dev*	Eval	Eval*
MFCCSDC-BASELINE	0.073	0.239	0.219
SSNN tanh610-tanh256	0.089	0.335	NA
FUSION	**0.045**	0.239	0.214

4. Results

The primary metric for all LRE15 results is *avgCavg*. It is an average of Cavg over the language clusters as defined in [13]. Two scores are computed for each system. *Dev** are scores on the development set, calibrated on this same set (i.e. cheating calibration) to show the lower bound for the cost function without any calibration error. *Eval* are scores on the original evaluation set calibrated on the development set.

Various experiments to determine optimal topology, sizes of layers and non-linearities were carried out. We were experimenting with layers from size 32 up to 1024 and with *tanh*, *sigmoid* and *ReLU* non-linearities. Different topologies were also tried - up to 2 hidden layers before summarization and up to 2 layers after summarization.

The best combination turned out to be one hidden layer with *tanh* non-linearity before summarization and one hidden linear layer after summarization. We were also trying to employ *L1*,*L2* and *Dropout* regularizations, without significant improvement in systems performance.

We found that the ideal amount of trainable parameters before summarization pseudo-layer should be around 300 000 per layer. This requirement gave us width of first layer fixed to 610 neurons. Determining the size of *summary vector* was the final step of experimentation — we can see different sizes and their influence on performance of the system on Fig. 3.

Table 4 shows, that our SSNN systems yield similar results to baseline, but still they are about 30% relatively worse. This situation changes when we fuse our best SSNN system with 256 dimensional *summary vector* and baseline system: the results improved by 30% relative in comparison with the baseline system on the development set. No change occurs for evaluation set performance. We speculate, this can be caused by wrongly estimated calibration. This theory is supported by the last column of Fig. 4 (*eval** scores); here we calibrated also evaluation scores on the eval set (i.e. cheating calibration). Some improvement is shown, but it is rather insignificant.

5. Conclusion

We have explored Sequence Summarizing Neural Networks (SSNNs) as an alternative to i-vector scoring in LID. On contrary to previsously published works on DNNs in LID, SSNN has the advantage of discriminative end-to-end training without any generative component or post-averaging of language posteriors. In SSNN, the summarization layer is responsible for

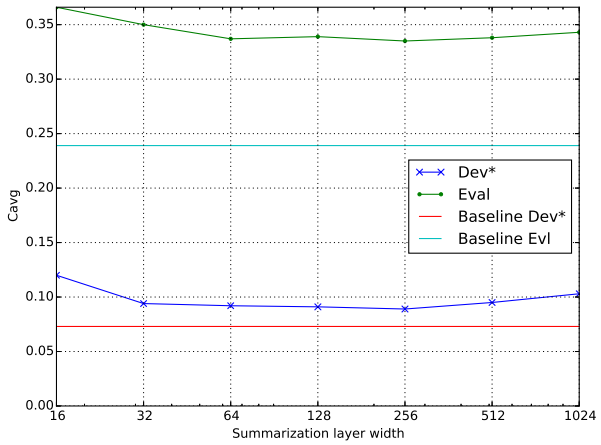

Figure 3: Performance with different sizes of *summary vectors*

converting variable-length feature sequence into one vector that is then forwarded through the rest of the DNN.

We have compared SSNN to a single state-of-the-art i-vector based baseline system based on standard MFCC/SDC features (intentionally omitting bottle-neck features requiring to be trained within an ASR system) and found that SSNN reaches comparable, though worse performance. In fusion, we have seen 30% relative improvement on the development set, but so far no positive change on the evaluation one.

In our future work, we will focus on the generalization of SSNN, its calibration properties (experiments with cheating calibration on the eval set suggest possible issues) and we will also test SSNN on more standard NIST LRE data from older evaluations. As DNNs are sensitive to training data amounts, we are especially interested in the performance of SSNN in conditions with abundant training data.

6. Acknowledgements

This work was supported by the DARPA RATS Program under Contract No. HR0011-15-C-0038. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. The work was also supported by Czech Ministry of Interior project No. VI20152020025 "DRAPAK", European Union's Horizon 2020 project No. 645523 BISON and Czech Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project "IT4Innovations excellence in science - LQ1602".

7. References

[1] H. Li, B. Ma, and K. A. Lee, "Spoken language recognition: from fundamentals to practice," *Proceedings of the IEEE*, vol. 101, no. 5, pp. 1136–1159, 2013.

[2] 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.

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

[4] 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.

[5] I. Lopez-Moreno, J. Gonzalez-Dominguez, and O. Plchot, "Automatic language identification using deep neural networks," in *Proc. ICASSP*, 2014.

[6] "2009 language recognition evaluation plan," http://goo.gl/5CHBLj.

[7] J. Gonzalez-Dominguez, I. Lopez-Moreno, H. Sak, J. Gonzalez-Rodriguez, and P. J. Moreno, "Automatic language identification using long short-term memory recurrent neural networks." in *INTERSPEECH*, 2014, pp. 2155–2159.

[8] K. Vesely, "Sequence summarizing neural network for speaker adaptation," in *ICASSP*, 2016, pp. 0–0.

[9] K. Zmolikova, M. Karafiat, K. Vesely, M. Delcroix, S. Watanabe, L. Burget, and J. Cernocky, "Data selection by sequence summarizing neural network in mismatch condition training," in *submitted to Interspeech*, 2016.

[10] R. Fér, P. Matějka, F. Grézl, O. Plchot, and J. Černocký, "Multilingual bottleneck features for language recognition," in *Proc. Interspeech 2015*, pp. 389–393.

[11] K. Vesely, M. Karafiát, F. Grezl, M. Janda, and E. Egorova, "The language-independent bottleneck features," in *Spoken Language Technology Workshop (SLT), 2012 IEEE*. IEEE, 2012, pp. 336–341.

[12] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[13] "2015 language recognition evaluation," http://goo.gl/lHNK0I.

[14] O. Plchot, P. Matějka, R. Fér, O. Glembek, O. Novotný, J. Pešán, K. Veselý, L. Ondel, M. Karafiát, F. Grézl, S. Kesiraju, L. Burget, N. Brümmer, A. Swart, S. Cumani, S. H. Mallidi, and R. Li, "Bat system description for nist lre 2015," *Proc. Odyssey-16, Bilbao, Spain*, 2016.

[15] P. A. Torres-Carrasquillo, E. Singer, M. A. Kohler, R. J. Greene, D. A. Reynolds, and J. R. Deller Jr, "Approaches to language identification using gaussian mixture models and shifted delta cepstral features." in *INTERSPEECH*, 2002.

[16] H. Hermansky and N. Morgan, "Rasta processing of speech," *Speech and Audio Processing, IEEE Transactions on*, vol. 2, no. 4, pp. 578–589, 1994.

[17] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude." COURSERA: Neural Networks for Machine Learning, p. 4, 2012.