Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020), pages 3566–3572
Marseille, 11–16 May 2020

On The Performance of Time-Pooling Strategies for End-to-End Spoken Language Identification

João Monteiro[1,2], **Jahangir Alam**[1,2], **Tiago H. Falk**[1]
[1]Institut National de la Recherche Scientifique (INRS-EMT), Quebec, Canada
[2]Centre de Recherche Informatique de Montréal (CRIM), Quebec, Canada
joao.monteiro@emt.inrs.ca, jahangir.alam@crim.ca, falk@emt.inrs.ca

Abstract

Automatic speech processing applications often have to deal with the problem of aggregating local descriptors (i.e., representations of input speech data corresponding to specific portions across the time dimension) and turning them into a single fixed-dimension representation, known as global descriptor, on top of which downstream classification tasks can be performed. In this paper, we provide an empirical assessment of different time pooling strategies when used with state-of-the-art representation learning models. In particular, insights are provided as to when it is suitable to use simple statistics of local descriptors or when more sophisticated approaches are needed. Here, language identification is used as a case study and a database containing ten oriental languages under varying test conditions (short-duration test recordings, confusing languages, unseen languages) is used. Experiments are performed with classifiers trained on top of global descriptors to provide insights on open-set evaluation performance and show that appropriate selection of such pooling strategies yield embeddings able to outperform well-known benchmark systems as well as previously results based on attention only.

Keywords: Spoken language identification, Temporal pooling, End-to-end identification

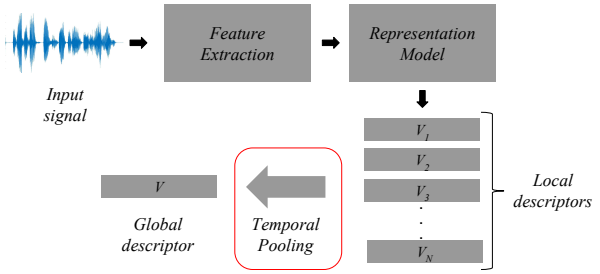


Figure 1: Mapping an input signal to a low-dimensional space. Pooling across the time dimension combines a set of local descriptors into a single global descriptor vector.

1. Introduction

Language identification (LID) from speech corresponds to the task of identifying the spoken language from a given speech example under the assumption that a single language is present. LID is commonly tackled using similar approaches as those employed for e.g., speaker verification/recognition. Representative examples include the use of low-dimensional language-dependent i-vectors (Dehak et al., 2011b), which are similar to speaker-dependent i-vectors used in speaker recognition (Dehak et al., 2011a). These are obtained by first computing a generative model of frame-level features, i.e., a universal background model (UBM), followed by factor analysis performed on top of statistics of latent variables in order to obtain a fixed-dimension representation that is independent of the input speech duration. Downstream tasks can then rely on these representations rather than general-purpose features.

In recent years, following the success of deep learning, generative modeling (Goodfellow et al., 2014) and deep neural networks have become a popular alternative to i-vectors. Commonly, such methods are employed to either map features obtained from speech recordings directly to low-dimensional representations (Zhong et al., 2017) or to perform recognition in an end-to-end fashion (Rohdin et al., 2017; Snyder et al., 2016), thus bypassing a second classification stage. Representative examples include the use of so-called x-vectors in speaker recognition (Snyder et al., 2017). In this case, feed-forward neural networks operating both at frame- and utterance-levels, perform statistical pooling of frame-level outputs so as to compute low-dimensional embeddings from utterances of varying lengths. Model training in this case is performed as a multi-class classification task over a closed-set of speakers; scoring, in turn, is performed using probabilistic linear discriminant analysis (PLDA) (Prince and Elder, 2007) using the output of an inner layer as low-dimensional embedding. More recently, other strategies based on artificial neural networks have also been applied directly to learn speaker-dependent representations using speaker recognition as a training task. In this case, convolutional neural networks have been proposed (Bhattacharya et al., 2017; Chung et al., 2018), relying on 2-dimensional convolutions over time-frequency speech representations. Specifically for LID, approaches introduced in (Cai et al., 2018a) and (Cai et al., 2018b) employ residual architectures and different strategies for training, such as center-loss minimization (Wen et al., 2016) and angular softmax (Liu et al., 2017), both inspired by applications in face recognition.

Figure 1 illustrates the generic framework used when applying neural networks for language-dependent representation learning. After feature extraction, a representation model maps these input features to a set of vectors which will be referred to as local descriptors. A global descriptor is lastly obtained by means of a temporal pooling strategy. In this work, we focus on extending our previous findings

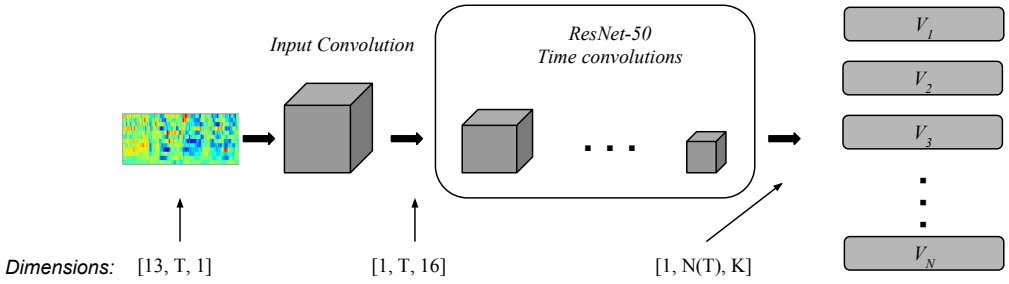


Figure 2: Block diagram: from input features to local descriptors.

(Monteiro et al., 2019) by assessing the performance on LID tasks of different strategies employed to perform pooling across the time dimension, as represented by the red box in Figure 1. We thus evaluate three such pooling methods ranging from the simple use of a linear projection of first- and second-order statistics to more complex learning-based approaches. Evaluation is carried out under different scoring strategies and varying conditions, such as using only short-duration recordings, evaluating trials corresponding to confusing languages, and including test recordings from languages not represented within training data.

The remainder of this paper is organized as follows: in Section 2 we provide a description of the modeling strategies employed. The experimental setup is presented along with results and discussion in Section 3, while conclusions are finally drawn in Section 4.

2. System description

2.1. Mapping speech into local descriptors

A block diagram describing the first part of our model, consisting of a mapping from features to local descriptors, is shown in Figure 2. The considered features were 13 Mel-frequency cepstral coefficients (MFCC), which are initially input into a convolutional layer consisting of 16 filters with dimension $[13, 3]$, thus shrinking the MFFCs dimension to 1. Following that, a temporal convolutional neural network is employed. We used the standard ResNet-50 and apply convolutions across the time dimension only. The convolutional stack's output is a set of N local descriptors $V_i \in \mathbb{R}^K$, where K is the number of filters in the last convolutional layer, set to 512 in our case. N is a function of the input length T.

2.2. Temporal pooling strategies

Once local descriptors are obtained, the following step consists of pooling them into a single global descriptor V. Next, we describe the three temporal pooling strategies evaluated herein:

1. *Statistics pooling*: Given a set of local descriptors $V_i \in \mathbb{R}^K, i \in \{1, 2, ..., N\}$, the global descriptor V will be given by a linear projection of concatenated element-wise estimates of first- and second-order statistics of V_i:

$$V = W[\mu(V_i) \wedge \sigma(V_i)], \tag{1}$$

where the linear transformation W has its entries learned jointly with the complete model. An illustration of the statistics pooling method is presented in Figure 4-(a). The projection W yields a final dimension of 128 on V.

2. *Attentive pooling*: We employ an attention scheme usually referred to as self-attention in order to weigh local descriptors in a data-dependent fashion prior to computing statistics, as indicated by the red block in Figure 4-(b). A linear transformation A, shared across time-steps i, is applied to each local descriptor V_i, resulting in the set of scalars $a_{1:N}$:

$$a_i = \tanh(AV_i). \tag{2}$$

A set of normalized weights summing up to 1 is then obtained through the softmax operator:

$$w_i = \frac{e^{a_i}}{\sum_{i=1}^{N} e^{a_i}}, \tag{3}$$

and the global descriptor V is given by the projection of concatenated statistics of weighted local descriptors:

$$V = W[\mu(w_i V_i) \wedge \sigma(w_i V_i)]. \tag{4}$$

The entries of A are learned along with the complete model. An illustration of the attentive pooling method is presented in Figure 4-(b).

3. *Recurrent attentive pooling*: The most complex pooling strategy considered herein consists in adding a recurrent block to the self-attention scheme described above. The recurrent model is implemented as a two-layered bi-directional LSTM (Hochreiter and Schmidhuber, 1997) with hidden layer set to a size of 256. The LSTM is used to: (i) summarize the set V_i into a pooled descriptor given by the hidden layer at the last time-step H, and (ii) map the sequence V_i into another sequence U_i, which provides an extra time modelling mechanism. The self-attention previously described is employed on top of U_i, and V is thus given by:

$$V = W[\mu(w_i U_i) \wedge \sigma(w_i U_i) \wedge H]. \tag{5}$$

Note that in this setting the same attentive strategy is performed on top of the sequence U_i output by the recurrent model. During the projection of statistics of weighted descriptors, H is further concatenated. An illustration of the recurrent attentive pooling method is presented in Figure 4-(c).

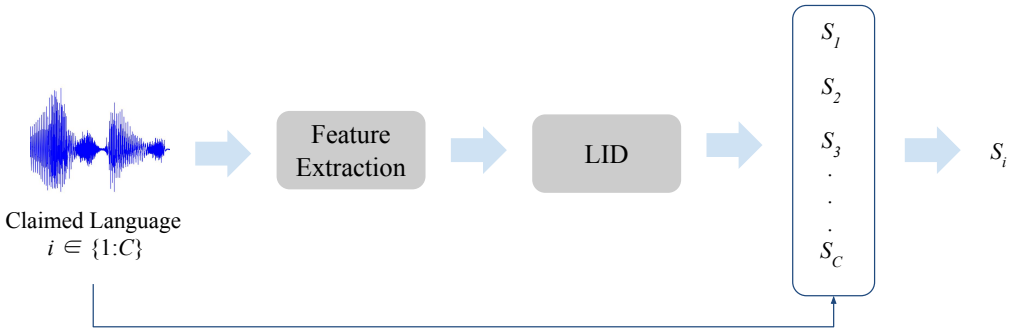


Figure 3: End-to-end scoring: the logit S_i corresponding to the claimed language i is returned as a verification score.

2.3. Training

Training is performed under the multi-class classification setting through minimization of the multi-class cross-entropy loss measured with an extra softmax output layer obtained through an affine projection of V. Additionally, discriminability of global descriptors V with respect to spoken language is further enforced through the minimization of the triplet loss T given by:

$$T = \text{softplus}(d_+ - d_-), \tag{6}$$

where d_+ and d_- correspond to the difference to 1 of the cosine similarity measures between pairs of global descriptors obtained from recordings of the same, and from different languages, respectively. The softplus operator is such that:

$$\text{softplus}(x) = \log(1 + e^x),\ x \in \mathbb{R}. \tag{7}$$

RMSProp (Tieleman and Hinton, 2012) is employed for updating model's parameters. Its smoothing constant is set at 0.99, while the global learning rate starts at 0.001 and is halved once the classification error rate, measured on a validation set held out of training, plateaus for 30 epochs. Minibatches of size 64 are constructed such that two random recordings of each language are sampled sequentially to form positive pairs, and a random recording from a different language is selected to compose the negative pair. One epoch is considered finished when each language is selected 1000 times to compose positive pairs. A budget of 500 epochs is used for each training run.

Recordings are further processed during training such that, for recordings longer than 6 seconds, a random continuous segment of 6 seconds duration is selected. Short-duration training recordings are elongated with initial frames so as to make them reach a minimum of 6 seconds duration, which is done by simply taking a random chunk of all examples across the minibatch. To further increase the diversity on training samples, each minibatch has its length randomly selected to lie between 3-6 seconds before feeding it to the neural network. At test time, recordings are processed as is, without any modification to their length[1].

[1]Code is available at: `https://github.com/joaomonteirof/e2e_LID`

Table 1: Dataset details for all languages.

	Train		*Evaluation*	
Language	*#Speakers*	*Utt./Speaker*	*#Speakers*	*Utt./Speaker*
Cantonese	24	320	6	300
Mandarim	24	300	6	300
Indonesian	24	320	6	300
Japanese	24	320	6	300
Russian	24	300	6	300
Korean	24	300	6	300
Vietnamese	24	300	6	300
Kazakh	86	50	86	20
Tibetan	34	330	34	50
Uyghur	353	20	353	5

3. Experiments

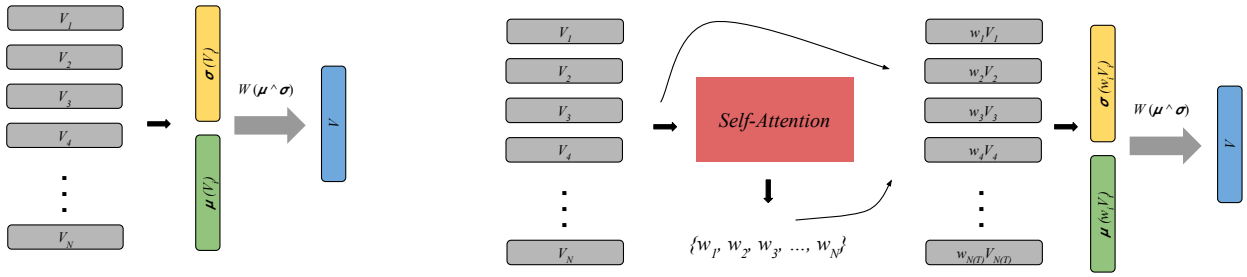
3.1. Data description

Experiments are performed on the dataset introduced for the AP18-OLR Challenge (Tang et al., 2018), consisting of recordings from 10 oriental languages. Train, development and evaluation data partitions were made available. We further introduce multi-condition training data by augmenting the original train partition with supplementary noisy speech, created by corrupting original recordings adding reverberation (reverberation time varies from 0.25s - 0.75s) and background noise such as music (signal-to-noise ratio, SNR, within 5-15dB), and babble (SNR within 10-20dB). Noise signals were taken from the MUSAN corpus (Snyder et al., 2015) and the room impulse responses used to simulate reverberation effects were the same as those introduced in (Ko et al., 2017). Further details regarding the employed data are shown in Table 1.

Three evaluation conditions are considered, namely:

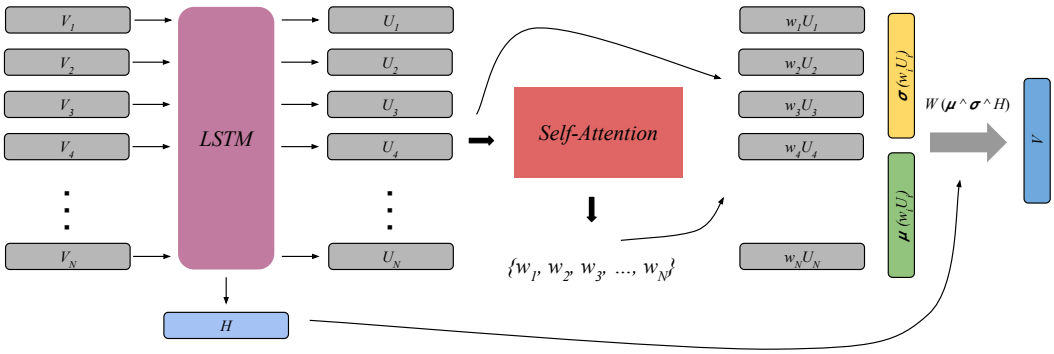
1. *Short-duration*: Only test recordings with less than one second are evaluated.
2. *Confusing languages*: Claimed languages and test recordings correspond to languages known to be confusing: Cantonese, Korean, and Mandarin.
3. *Unseen languages*: Test recordings from languages other than the ones represented within train data are included.

A total of 214560, 22071, and 404160 trials were made available for each of the evaluation conditions. By trial we mean a pair *claimed language/test recording*, and the test recording might or might not correspond to the claimed



(a) Statistics pooling

(b) Attentive pooling



(c) Recurrent attentive pooling

Figure 4: Different pooling strategies considered.

language, in which case trials are labeled target or non-target, respectively. For the case of trials corresponding to the third evaluation condition, non-target trials trials are added in which case claimed languages are one of the 10 in Table 1 while the test recording correspond to an out-of-distribution language, never presented to the model at train time, and even so, models should be able to correctly classify such a trial as non-target.

3.2. Evaluation

Two metrics are employed for assessing performance of studied models: the equal error rate (EER) and average cost performance (C_{avg}). EER consists of the value of the false acceptance rate at the threshold in which it matches the false rejection rate, while C_{avg} averages the missing and false alarm probabilities for each possible target/non-target pair of languages. More details about both metrics can be found in (Tang et al., 2018). We remark all reported results in terms of both EER and C_{avg} were computed with the official scripts released for the AP18-OLR challenge.

We start our evaluation by comparing performances of the described models under an end-to-end scoring strategy. We do so by simply using the output of the softmax layer corresponding to the claimed identity as a score for a given trial, i.e. the softmax output corresponding to how likely it is that a given recording belongs to a particular class is directly used as a score indicating how likely a match between the test recording and the claimed language is. Such score scheme is depicted in Figure 3. For performance baseline, an extra model is evaluated corresponding to a 6-layered convolutional neural network followed by a 2-layered bi-directional LSTM. Training in the case of the baseline model is performed with the same approach as described previously. By inspecting results reported in Table 2 for the short-duration case (Eval. 1), one can notice that simple statistics pooling yields better identification performance when compared to more complex methods which rely on longer term dependencies. Statistics pooling and the recurrent attentive setting present approximately the same performance on the other evaluation conditions.

Next, we evaluate our models using scoring strategies which would be valid in an open-set condition, i.e. if at test time recordings from languages not presented to the model during training were observed. We thus employ the cosine similarity as well as PLDA to score trials considering as enrollment language models the average of global descriptors V obtained over all test recordings of a given language. Well-known i-vectors are included as a benchmark and are obtained by computing a 512-Gaussians full covariance UBM on the training partition using the same MFCC features as our models. Total variability analysis is performed on top of UBM's Baum-Welch statistics in order to obtain a 400-dimensional i-vector extractor.

The first conclusion one can draw from results reported in Table 3 is that the proposed method is always able to outperform i-vectors by a large difference in both scoring strategies considered. Moreover, for the case of cosine scoring the recurrent attentive pooling attains the best per-

Table 2: End-to-end evaluation.

	Eval. 1		*Eval. 2*		*Eval. 3*	
	EER (%)	C_{avg}	EER (%)	C_{avg}	EER (%)	C_{avg}
Conv-LSTM	24.00	0.2321	7.54	0.0738	7.57	0.0491
Statistics	**10.85**	**0.1122**	3.63	0.0358	**4.23**	**0.0200**
Attennntion	12.62	0.1246	6.80	0.0669	5.65	0.0315
LSTM+Att.	13.11	0.1285	**3.41**	**0.0335**	4.25	0.0222

Table 3: Cosine similarity and PLDA evaluation.

		Eval. 1		Eval. 2		Eval. 3	
		EER (%)	C_{avg}	EER (%)	C_{avg}	EER (%)	C_{avg}
Cosine	i-vector	18.02	0.1780	10.71	0.1069	7.77	0.0577
	Statistics	14.52	0.1418	11.40	0.1071	5.25	0.0405
	Attention	14.63	0.1432	9.81	0.0967	6.44	0.0463
	LSTM+Att.	**12.43**	**0.1216**	**3.89**	**0.0378**	**4.51**	**0.0284**
PLDA	i-vector	17.50	0.1743	10.66	0.1059	7.51	0.0524
	Statistics	**11.22**	**0.1104**	4.15	0.0419	**4.15**	**0.0226**
	Attention	13.48	0.1328	8.28	0.0810	5.97	0.0369
	LSTM+Att.	15.36	0.1455	**3.37**	**0.0334**	6.93	0.0268

formance among compared approaches. For PLDA, on the other hand, that is only the case for the evaluation with confusing languages (Eval. 2). Moreover, by comparing results across Tables 2 and 3, we observe that end-to-end evaluation of the statistics pooling approach performs better (Eval. 1 and 2) or matches (Eval. 3) the performance obtained when scoring with PLDA. For the recurrent attentive pooling, in turn, cosine similarity and PLDA yield the best results. End-to-end scoring attains the lowest EER across all compared scoring methods when non-target unseen languages are included (Eval. 3). Self-attentive pooling, on the other hand, achieves its better performance with end-to-end scoring in all evaluation conditions.

Additionally, we evaluate the effect of pretraining the convolutional layers while using the simplest pooling strategy, and later fine-tuning a more complex temporal pooling by initializing the convolutional blocks with the weights and biases provided by the previously trained model and retraining utilizing the same train data. This is done with the goal of giving focus during the second training phase to parameters related to the pooling modules since convolution weights are initialized in a well-performing setting. Such results are reported in Table 4 in which the performance for each of the three evaluation conditions are compared with all of the scoring strategies considered.

In most observed cases, pretraining with a simple pooling strategy is observed to boost performance when compared to completely training the convolutional blocks and the pooling part altogether from scratch. The performance improvement is particularly observed in the case of end-to-end evaluation. Overall, when comparing results across Tables 2- 4, we observe that simple statistics pooling to be the best performing model for short-duration recordings, while in the case of confusing languages and trials containing test recordings from unseen languages, the recurrent attentive pooling was the one to achieve the lowest EER after fine-tuning a model pretrained with statistics pooling. In the case of cosine and PLDA scoring, pretraining does not always help improving performance. In fact, for such cases the fine tuning is observed to actually degrade the EER of previously trained models.

Finally, we present in Table 5 the identification performance of our best systems, i.e. those pre-trained with simple pooling strategies and then retrained after adding a complex pooling scheme, alongside baselines systems for a clear side-by-side comparison with other well-known approaches with different backends (or no backend at all for the end-to-end cases).

The baseline systems correspond to the i-vector setting discussed in Table 3, evaluated with both cosine similarity and PLDA, as well as the convolutional/recurrent end-to-end approach used as a reference in Table 2, but this time cosine and PLDA evaluation is further reported for those systems. We additionally include what we refer to as *Tandem*, consisting of principal components of Baum-Welch statistics obtained from a GMM-UBM trained on top of tandem features, in a similar setting to that described in (Alam et al., 2016). In our case, however, we train the UBM with MFCCs augmented with the phonetic posterior distribution output by an acoustic model trained in advance. The acoustic model was implemented as a neural network trained on the THCHS30 corpus, an open Chinese speech database (Dong Wang, 2015).

Results corroborate our previous findings in that our proposed systems outperform the baselines by a significant difference in all considered conditions. We remark that the additional phonetic information appears to be helpful in the short-duration case since the system based on tandem features outperformed i-vectors in that evaluation, which indicates such space might useful for training systems such as the ones proposed herein. We intend to investigate that in future work.

Table 4: Effect of pretraining (labeled as "*Pre*" in the table) on system performance.

		Eval. 1		*Eval. 2*		*Eval. 3*	
		EER (%)	C_{avg}	EER (%)	C_{avg}	EER (%)	C_{avg}
End-to-end	Attention	12.62	0.1246	6.80	0.0669	5.65	0.0315
	Pre+Attention	**10.97**	**0.1108**	**4.34**	**0.0427**	**4.58**	**0.0226**
	LSTM+Att.	13.11	0.1285	3.41	0.0335	4.25	0.0222
	Pre+LSTM+Att.	**11.76**	**0.1149**	**3.34**	**0.032**	**4.00**	**0.0206**
Cosine	Attention	**14.63**	0.1432	**9.81**	**0.0967**	6.44	0.0463
	Pre+Attention	14.64	**0.1418**	11.70	0.1105	**5.78**	**0.0451**
	LSTM+Att.	**12.43**	**0.1216**	**3.89**	**0.0378**	**4.51**	**0.0284**
	Pre+LSTM+Att.	14.83	0.1376	4.68	0.0422	4.88	0.0334
PLDA	Attention	**13.48**	**0.1328**	**8.28**	**0.0810**	**5.97**	**0.0369**
	Pre+Attention	17.63	0.1687	20.80	0.1976	20.61	0.0993
	LSTM+Att.	15.36	0.1455	**3.37**	**0.0334**	6.93	0.0268
	Pre+LSTM+Att.	**12.99**	**0.126**	3.95	0.0392	**4.37**	**0.0240**

Table 5: Performance comparison of proposed system (last six rows) and benchmarks based on equal error rate (%) and average cost performance (C_{avg}).

		Eval. 1		*Eval. 2*		*Eval. 3*	
		EER (%)	C_{avg}	EER (%)	C_{avg}	EER (%)	C_{avg}
Benchmarks	i-vector+Cosine	18.02	0.1780	10.71	0.1069	7.77	0.0577
	i-vector+PLDA	17.50	0.1743	10.66	0.1059	7.51	0.0524
	Tandem+Cosine	15.73	0.1502	13.81	0.1387	8.98	0.0683
	Tandem+PLDA	15.30	0.1461	13.33	0.1324	8.37	0.0596
	Conv-LSTM+Cosine	20.10	0.1978	9.11	0.0840	7.78	0.0537
	Conv-LSTM+PLDA	19.14	0.1896	8.78	0.0819	7.49	0.0490
	Conv-LSTM	24.00	0.2321	7.54	0.0738	7.57	0.0491
Proposed - Cosine	Statistics	14.52	0.1418	11.40	0.1071	5.25	0.0405
	Pre+Attention	14.64	0.1418	11.70	0.1105	5.78	0.0451
	Pre+LSTM+Att.	14.83	0.1376	4.68	0.0422	4.88	0.0334
Proposed - PLDA	Statistics	11.22	0.1104	4.15	0.0419	4.15	0.0226
	Pre+Attention	17.63	0.1687	20.80	0.1976	20.61	0.0993
	Pre+LSTM+Att.	12.99	0.126	3.95	0.0392	4.37	0.0240
Proposed - End-to-End	Statistics	**10.85**	0.1122	3.63	0.0358	4.23	0.0200
	Pre+Attention	10.97	**0.1108**	4.34	0.0427	4.58	0.0226
	Pre+LSTM+Att.	11.76	0.1149	**3.34**	**0.0320**	**4.00**	**0.0206**

4. Conclusion

In this work we tackled the problem of language identification from speech. Under this setting, most of the approaches based on artificial neural networks can be described with two main components: (i) a representation model which is intended to map speech features into local descriptors, i.e. vectors representing parts of the input signal across the time dimension, and (ii) a temporal pooling strategy aimed at combining such local descriptors into a single representation vector. We are particularly interested in evaluating and comparing the identification performance of pooling strategies stacked over a sequence of 1-dimensional convolutional layers across time. Experiments are carried out using the data provided for the AP18-OLR challenge which contains recordings from 10 oriental languages, and further provides 3 challenging evaluation conditions.

Given that our considered models are able to outperform well-known benchmark systems, we narrow our attention to empirically evaluating differences in identification performance given varying evaluation conditions as well as scoring strategies. Our main conclusion is that the most suitable pooling strategy depends on the evaluation setting, as simple statistical pooling yielded the best performance when short-duration test utterances are used at test time, while more sophisticated strategies such as the recurrent attentive scheme yielded the best performance with confusing languages and unseen non-target trials. Moreover, all the best observed identification performances were attained under end-to-end scoring rather than with the use of complex schemes such as PLDA. Pretraining with simple pooling strategies and then fine tuning a model after adding pooling blocks was also observed to improve performance.

For future investigation, we believe repeating the experimental setting proposed here for automatic speaker verification problems would be insightful, and hence we envision doing so under varying evaluating conditions such as across different languages between train and evaluation data.

5. Acknowledgements

The authors wish to acknowledge funding from the National Research Council of Canada (NRC) through the Canadian Indigenous Languages Technology project under contract 909859, and from the Natural Sciences and Engineering Research Council of Canada (NSERC) through contract/grant RGPIN-2016-4175, and RGPAS-493010-2016. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the NRC and NSERC.

6. References

Alam, M. J., Kenny, P., and Gupta, V. (2016). Tandem features for text-dependent speaker verification on the reddots corpus. In *INTERSPEECH*, pages 420–424.

Bhattacharya, G., Alam, J., and Kenny, P. (2017). Deep speaker embeddings for short-duration speaker verification. In *Proc. Interspeech 2017*, pages 1517–1521.

Cai, W., Cai, Z., Liu, W., Wang, X., and Li, M. (2018a). Insights into end-to-end learning scheme for language identification. *arXiv preprint arXiv:1804.00381*.

Cai, W., Chen, J., and Li, M. (2018b). Exploring the encoding layer and loss function in end-to-end speaker and language recognition system. *arXiv preprint arXiv:1804.05160*.

Chung, J. S., Nagrani, A., and Zisserman, A. (2018). Voxceleb2: Deep speaker recognition. In *INTERSPEECH*.

Dehak, N., Kenny, P. J., Dehak, R., Dumouchel, P., and Ouellet, P. (2011a). Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(4):788–798.

Dehak, N., Torres-Carrasquillo, P. A., Reynolds, D., and Dehak, R. (2011b). Language recognition via i-vectors and dimensionality reduction. In *Twelfth annual conference of the international speech communication association*.

Dong Wang, Xuewei Zhang, Z. Z. (2015). Thchs-30 : A free chinese speech corpus.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Ko, T., Peddinti, V., Povey, D., Seltzer, M. L., and Khudanpur, S. (2017). A study on data augmentation of reverberant speech for robust speech recognition. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5220–5224. IEEE.

Liu, W., Wen, Y., Yu, Z., Li, M., Raj, B., and Song, L. (2017). Sphereface: Deep hypersphere embedding for face recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, page 1.

Monteiro, J., Alam, J., and Falk, T. H. (2019). Residual convolutional neural network with attentive feature pooling for end-to-end language identification from short-duration speech. *Computer Speech & Language*.

Prince, S. J. and Elder, J. H. (2007). Probabilistic linear discriminant analysis for inferences about identity. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE.

Rohdin, J., Silnova, A., Diez, M., Plchot, O., Matejka, P., and Burget, L. (2017). End-to-end DNN Based Speaker Recognition Inspired by i-vector and PLDA. *ArXiv e-prints*, October.

Snyder, D., Chen, G., and Povey, D. (2015). MUSAN: A Music, Speech, and Noise Corpus. arXiv:1510.08484v1.

Snyder, D., Ghahremani, P., Povey, D., Garcia-Romero, D., Carmiel, Y., and Khudanpur, S. (2016). Deep neural network-based speaker embeddings for end-to-end speaker verification. In *Spoken Language Technology Workshop (SLT), 2016 IEEE*, pages 165–170. IEEE.

Snyder, D., Garcia-Romero, D., Povey, D., and Khudanpur, S. (2017). Deep neural network embeddings for text-independent speaker verification. *Proc. Interspeech 2017*, pages 999–1003.

Tang, Z., Wang, D., and Chen, Q. (2018). Ap18-olr challenge: Three tasks and their baselines. *arXiv preprint arXiv:1806.00616*.

Tieleman, T. and Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31.

Wen, Y., Zhang, K., Li, Z., and Qiao, Y. (2016). A discriminative feature learning approach for deep face recognition. In *European Conference on Computer Vision*, pages 499–515. Springer.

Zhong, J., Hu, W., Soong, F., and Meng, H. (2017). Dnn i-vector speaker verification with short, text-constrained test utterances. *Proc. Interspeech 2017*, pages 1507–1511.