INTERSPEECH 2021
30 August – 3 September, 2021, Brno, Czechia

Language recognition on unknown conditions: the LORIA-Inria-MULTISPEECH system for AP20-OLR Challenge

Raphaël Duroselle, Md Sahidullah, Denis Jouvet, Irina Illina

Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France
firstname.name@loria.fr

Abstract

We describe the LORIA-Inria-MULTISPEECH system submitted to the Oriental Language Recognition AP20-OLR Challenge. This system has been specifically designed to be robust to unknown conditions: channel mismatch (task 1) and noisy conditions (task 3). Three sets of studies have been carried out for elaborating the system: design of multilingual bottleneck features, selection of robust features by evaluating language recognition performance on an unobserved channel, and design of the final models with different loss functions which exploit channel diversity within the training set. Key factors for robustness to unknown conditions are data augmentation techniques, stochastic weight averaging, and regularization of TDNNs with domain robustness loss functions. The final system is the combination of four TDNNs using bottleneck features and one GMM using SDC-MFCC features. Within the AP20-OLR Challenge, it achieves the top performance for tasks 1 and 3 with a C_{avg} of respectively 0.0239 and 0.0374. This validates the approach for generalization to unknown conditions.

Index Terms: language recognition, channel mismatch, domain generalization

1. Introduction

Language recognition is the task of predicting the language used in a test speech utterance [1]. State-of-the-art language recognition systems are based on the discriminative training of a deep neural network [2] with multilingual bottleneck features [3]. They rely on automatic training and consequently highly depend on the training dataset. A drop in performance is expected from a domain shift of the test data, such as noise or channel mismatch [4]. Various approaches have been introduced to reduce the impact of domain shift over language recognition systems: design of robust features [5, 6], data augmentation of the training set [5, 7], feature-based adaptation [8] and model-based adaptation of the classifier [6, 7, 9, 10].

The Oriental Language Recognition (OLR) Challenge has been organized for the last five years with the goal of boosting research on language recognition technology [11, 12, 13, 14]. The AP20-OLR edition [15] focuses on three tasks: cross channel language identification (task 1), dialect identification (task 2) and noisy language identification (task 3). The LORIA-Inria-MULTISPEECH team focused on improving robustness to unknown conditions in order to compete for task 1 (cross-channel identification) and task 3 (noisy conditions). We did not participate in task 2. This paper summarizes our research effort towards the design of a robust language recognition recipe suited for domain generalization.

Three sets of studies have been carried out to design a robust language recognition system. First, we designed a robust frame-level *bottleneck features* extractor, with a recipe based on an end-to-end speech recognition model. Then, in order to simulate mismatched test conditions, we compared several utterance-level classifiers trained with different recipes, without using training data from the unknown channels, and evaluated their performance on unknown channels. In the paper, we call this step *optimization for system robustness*. Finally, we designed a system combination strategy. We proposed to train several models on a large dataset including data from unknown channels, and merged their prediction in order to increase robustness. In the following, this step is called the *final system design*. For each step specific training and testing sets have been used.

The submitted system is the same for task 1 and 3. It is constituted of the fusion of five models: one GMM model and four TDNNs, trained with bottleneck features and different loss functions. We propose a new recipe for training multilingual bottleneck features from a *Conformer* model [16], artificial data augmentation techniques abiding by the rules of the challenge and stochastic weight averaging [17] to increase generalization of the model. Other key factors that have been investigated are the combination of models with different properties and a duration-dependent calibration [18].

2. Dataset usage

2.1. Corpora

The rules of the challenge specify a limited set of corpora, which can be used to develop the systems [15].The training data contains 16 languages. The 10 traditional languages of the OLR challenges are Cantonese (ct-cn), Mandarin (zh-cn), Indonesian (id-id), Japanese (ja-jp), Russian (ru-ru), Korean (ko-kr), Vietnamese (vi-vn), Kazakh (ka-cn), Tibetan (ti-cn), and Uyghur (uy-cn). Only these languages are provided with transcriptions in the sets AP16-OL7 and AP17-OL3. In addition, recordings for three Chinese dialects are provided: Hokkien, Sichuanese, and Shanghainese. Moreover, data from three non target languages of previous evaluations are provided: Catalan, Greek, and Telugu.

The sets containing recordings from unknown channels are of key interest: AP19-OLR-dev-task2 and AP19-OLR-test-task2. They only contain 6 languages: Japanese, Russian, Vietnamese, Tibetan, Mandarin and Uyghur. In the paper, they are referred to as the *development languages*. This means that we do not have access to recordings from unknown channels for three of the six languages of task 1: Cantonese, Indonesian and Korean. Consequently, we designed a two-step strategy to use this data from unknown channels first for evaluating performance on unknown channels (*optimization for system robustness*) and then to increase the diversity of the training set (*final system design*).

As mentioned before, the system has been designed in several steps. Train, validation and test datasets used for each step are described in Table 1 and explained below:

http://dx.doi.org/10.21437/Interspeech.2021-276

Table 1: *Usage of datasets for each set of experiments.* val. *refers to validation.*

Dataset	*bottleneck features*	*optimization for system robustness*	*final system design*
AP16-OL7	train & val.	train & val.	train & val.
AP17-OL3	train & val.	train & val.	train & val.
AP17-test		train & val.	train & val.
AP18-test		test-2018	train & val. & match
AP19-dev-task2		dev-2019	mismatch
AP19-dev-task3		dev-2019	train & val.
AP19-test-task1		test-2019	train & val. & mismatch
AP19-test-task2		test-2019	train & val.
AP19-test-task3		test-2019	train & val.
AP20-dialect			train & val.

1. Training of *bottleneck features*. We use recordings from the ten traditional languages with their transcriptions.
2. *Optimization for system robustness*. In order to replicate the unknown channel evaluation conditions, we design systems using data from mobile channels only. We evaluate them on three datasets: *test-2018* (with mobile channel data), *dev-2019* and *test-2019* (with unknown channel data). To focus on the channel mismatch problem, we only use the 6 *development languages*.
3. *Final system design*. At the final stage of training, once we have selected training recipes robust to channel mismatch, we increase further diversity of the training set by adding data from unknown channels. We train the system with 16 languages. This allows us to evaluate the system for three different sets of languages: *development languages*, languages of task 1 and languages of task 3. We use two test sets: *match* (with only mobile channel data) and *mismatch* (with unknown channel data).

When an original dataset is split into several datasets for our experiments, we use speaker labels (when available) to ensure that all recordings of the same speaker belong to the same set.

2.2. Data augmentation

To improve the robustness of the systems to mismatched conditions, we train them with three data augmentation techniques. To abide by the rules of the challenge [15], we do not use any additional speech corpus and only use other files of the training corpus or artificial noise or filters. The three data augmentation methods are: addition of white noise, addition of babble noise (sampled by mixing other files of the training set) and convolution with random artificial band-pass filters.

3. Frame-level features

We explored two kinds of frame-level features during the challenge: spectral features and bottleneck features (BNF). We also used specific SDC-MFCC features for the GMM model.

3.1. Spectral features

Mel-filterbank features and mel-frequency cepstral coefficients (MFCCs) were evaluated. When used without any domain robustness technique, restriction to the telephone bandwidth (300-3800 Hz) was shown useful. No significant gain was achieved by adding pitch features [19].

The Gaussian mixture model (GMM) system employs shifted delta coefficients (SDC) computed with short-term MFCC features [20]. The MFCCs are extracted from speech frames of 20 ms with shift of 50% and 16 filters in mel scale. We process the extracted MFCCs with relative spectral (RASTA) processing for removing slowly varying channel effect. Then we compute SDCs with shift of three frames and a context of seven frames to create 128-dimensional SDC-MFCC features.

For all systems, we applied an energy-based speech activity detector (SAD) to discard the non-speech frames. Finally, we found utterance-level cepstral mean and variance normalization (CMVN) helpful for robustness of the TDNN models.

3.2. Bottleneck features

Multilingual bottleneck features [3] are very efficient frame-level features for language recognition. Usually, forced alignment between frame-level acoustic features and transcriptions is performed by an automatic speech recognition system for each target language. Then simple neural networks are trained to predict for each frame the assigned phone or triphone. An embedding is extracted from an intermediate bottleneck layer of the system.

In this work, we did not invest resources to develop an acceptable automatic speech recognition system for each target language with the goal of performing forced phone alignment. We simply trained a unique multilingual end-to-end speech recognition system with the connectionist temporal classification (CTC) loss [21], extending the idea of [22, 23]. We used the *Conformer* architecture [16] with an output layer specific to each target language, with 64 mel-filterbank features as input. The ten traditional languages were used for training the *Conformer* model. A sentence piece model [24] was estimated on the training transcriptions to define 2000 target tokens for each language. Finally we extracted frame-level embeddings from a hidden layer of the *Conformer* model and used them as language recognition features. We used a small *Conformer* encoder constituted of two blocks with four attention heads.

4. Neural network training

The utterance-level classification task can be performed by a neural network. The standard time delay neural network (TDNN) architecture for language recognition [2] was used in this work. First the general training recipe was selected according to experiments with the datasets defined for *optimization for system robustness*. For this set of experiments, we kept data from unkown channels for evaluation of the system. Then several systems were trained using the datasets defined for the *final system design*, with regularization loss functions that benefit from training data from unknown channels.

4.1. Training recipe

The final training recipe includes:

- the cross-entropy (CE) loss function
- stochastic gradient descent with dropout [25]
- specAugment [26]
- the three data augmentation techniques (Subsection 2.2)
- stochastic weight average [17]. It consists in averaging parameters of the model along the trajectory of the gradient descent, instead of selecting the parameters corresponding to the best validation loss.

4.2. Exploring other loss functions

In the last set of experiments, we used the *final system design* datasets with training data from unknown channels. Consequently, we trained systems with different loss functions to reduce the domain mismatch:

- additive angular margin softmax (AAM) [27] as an alternative classification loss function , we used a margin parameter $m = 0.1$ and a radius $s = 40$
- regularization of the cross-entropy with maximum mean discrepancy (MMD) between mobile channel and unknown channels [10], we compared different weights λ for the regularization loss functions
- regularization of the cross-entropy with n-pair loss [28]

5. Gaussian mixture model training

As an alternative utterance-level model, we chose the GMM-based model as this gives promising language recognition accuracy for short test utterances [1]. This statistical approach was also traditionally used in NIST language recognition evaluations (LREs) and could be helpful by providing complementary information to the neural network based methods. Language-dependent GMMs are trained using 4096 mixture components and ten iterations of the *expectation-maximization* (EM) algorithm. During scoring, we compute the log-likelihoods corresponding to each target language model separately.

6. Fusion and calibration

Standard linear multi-class calibration and score fusion are performed for each task with the FoCal toolkit [29]. This step is performed on the validation set defined for the *final system design*. The final scores are log-likelihood ratio.

For the selected final combination of systems, we implement a duration-dependent calibration procedure [18]. For three ranges of duration (inferior to 2s, between 2 and 4s, superior to 4s), we learn specific fusion and calibration parameters. For test utterances, the duration is estimated thanks to the energy-based SAD module mentioned in Section 3.

7. Experiments

In this section, we report the main results that lead to the design of the submitted system.

7.1. Bottleneck features training

We use the *bottleneck features* datasets. Models are compared with the average CTC loss for the 10 target languages on the validation set, cf. Table 2. We observe that the combination of cepstral mean and variance normalization (CMVN), specAugment and data augmentation allows to train better speech recognition features. We call these features *final BNF*, as opposed to the *baseline BNF*. We use the *final BNF* in the submitted system.

Table 2: *Automatic speech recognition performance in terms of CTC loss for different training recipes*

System name	Training recipe	CTC-loss on validation set
baseline BNF	mel-filterbank features	3.94
	+ CMVN	3.69
	+ specAugment	3.52
final BNF	+ data augmentation	**3.02**

7.2. Optimization for system robustness experiments

During the *optimization for system robustness* experiments, we train language recognition systems without using data from unknown channels. Performance is measured with the equal error rate (EER) on three test sets: *test-2018* (mobile channel), *dev-2019* and *test-2019* (which contain recordings from unknown channels), with the 6 *development languages*, cf. Table 3. We use a TDNN with the recipe described in Section 4, with different frame-level features and model selection strategies.

The experiments demonstrate the superiority of the trained bottleneck features over mel-filterbanks and MFCCs. *Final BNF* are superior to *baseline BNF* for the challenging *dev-2019* set. Finally, stochastic weight averaging is superior to model selection based on the best validation loss. We observed the same behavior for all models. In the following, stochastic weight averaging is applied to all models.

7.3. Final system design experiments

To design the final system we train various models with a large dataset containing data from 16 languages, including recordings from unknown channels. We calibrate them on the corresponding validation datasets and evaluate them on two datasets in terms of C_{avg} [15]: *match* (with mobile channel data only), *mismatch* (with some unknown channel recordings). Evaluation is performed for two sets of languages corresponding to task 1 and task 3. Results are presented in Table 4.

For comparison, we also present the performance obtained with a TDNN trained with cross-entropy loss on the *optimization for system robustness* datasets. All other models are trained with the *final system design* datasets and achieve better performance. Among the individual models, the best performance on unknown conditions is achieved by the TDNN trained with AAM loss. TDNNs trained with regularization loss functions (MMD and n-pair) do not improve on the test sets but the performance gap between known and unknown channels is reduced. Finally, the GMM model achieves the best performance on matched conditions but is very sensitive to channel mismatch.

Then, starting with the model with best performance, we greedily add models to the final system.We select the combination of five systems: four TDNNs based on BNF trained with additive angular margin softmax, regularization with maximum mean discrepancy with low and high weight values (λ), regularization with n-pair loss and one GMM model.

For known conditions, the best performance of individual systems is achieved by the GMM model (2.53% for task 1). Fusion with the TDNN trained with AAM helps to improve performance (1.54 %) but the addition of three other TDNNs trained with domain robustness objectives does not improve further (1.53%). Conversely, for unknown conditions, the TDNN trained with AAM is better than the GMM model (5.34% instead of 10.82% for task 1). The fusion of both is also helpful and the fusion with three other robust models allows an additional gain in performance (3.67%). Even if the models trained with regularization loss functions achieve a weak individual performance, they are designed to be robust to channel mismatch and are therefore useful on mismatched conditions for fusion with the best models.

Finally, we learn the fusion and calibration parameters for three different duration ranges. At test time, we use the fusion parameters corresponding to the speech duration of the test utterance. Since we had no information about durations of the evaluation utterances, we assumed that this technique was safer, even thought it was not beneficial for our test sets.

Table 3: *Language recognition performance for* system robustness *experiments. Equal error rate (%) on test sets. All models are TDNNs trained with the cross-entropy loss and data augmentation.*

input features	Model selection	test-2018	dev-2019	test-2019
mel-filterbanks	best validation loss	11.95	25.48	29.54
MFCCs	best validation loss	4.75	37.78	24.34
baseline BNF	best validation loss	3.91	19.11	**11.08**
final BNF	best validation loss	3.43	17.24	13.68
final BNF	stochastic weight average	**2.97**	**16.92**	12.78

Table 4: *Comparison of final systems in terms of C_{avg}. Performance is measured on the three development sets of the* final system design *step (validation, match and mismatch) and the evaluation sets (eval). The submitted system corresponds to the last row. It achieves an EER of* 2.47% *on task 1 and* 4.07% *on task 3.*

	models	Task 1 (6 languages) $C_{avg} \times 100$				Task 3 (5 languages) $C_{avg} \times 100$			
		val.	match	mismatch	eval	val.	match	mismatch	eval
individual models	pytorch x-vector baseline [15]				13.21				7.15
	CE, *optimization for system robustness* datasets	3.94	5.60	10.30	**4.96**	3.48	5.91	9.25	**4.02**
	CE, *system design* datasets	**2.50**	4.69	7.26	5.12	**2.04**	4.34	7.16	**4.94**
	AAM	3.68	3.14	**5.34**	**5.08**	3.87	3.20	**6.25**	5.11
	n-pair	3.22	5.73	7.25	7.52	2.58	5.14	7.36	5.70
	MMD $\lambda = 1$	3.18	5.39	7.87	5.54	2.93	4.99	8.31	5.58
	MMD $\lambda = 100$	4.01	5.89	7.60	7.61	3.76	5.24	7.29	5.80
	GMM	3.31	**2.53**	10.82	7.13	3.00	**2.69**	13.97	9.32
fusion	AAM - GMM	1.37	1.54	4.37	2.82	1.35	**1.45**	5.73	4.93
	AAM - GMM - MMD 100	0.98	1.57	3.91	2.47	1.00	1.60	4.96	3.82
	AAM - GMM - MMD 100 - n-pair - MMD 1	**0.93**	**1.53**	**3.67**	**2.28**	**0.97**	1.60	**4.54**	**3.71**
	duration-dependent calibration		1.56	3.82	2.39		1.61	4.60	3.74

8. Analysis of the submitted system

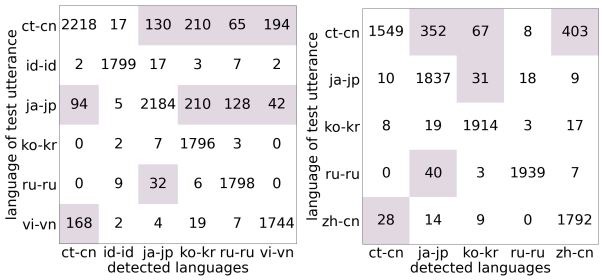
Once labels of the evaluation sets have been released, we notice that performances on the evaluation test sets of the systems trained with regularization loss functions are consistent with performance on the *mismatch* set (see Table 4). The two systems trained with the cross-entropy loss achieve a good individual performance, presumably because of matched conditions between the evaluation and training sets. The duration dependent calibration method does not improve performance.

For both tasks we perform an analysis of the errors according to the duration and estimations of signal to noise ratio (SNR) and pitch. Unsurprisingly, we observe that performance improves with duration and SNR. In addition, the submitted system is more efficient for high pitch values (200-250 Hz) than low pitch values (100-150 Hz).

Finally, for both tasks the errors are dominated by a few language pairs, as can be observed on the confusion matrices of Figure 1. A significant improvement in terms of C_{avg} can be expected by focusing on these specific language pairs.

9. Conclusion

The Oriental Language Recognition AP20-OLR Challenge was a good opportunity to evaluate the domain generalization ability of training recipes of language recognition systems. In this work, we propose a two-step strategy with the goal of optimally using data from unknown channels during design of the system. First, we design a robust recipe with a training set which does not contain data from unknown channels and keep this data from unknown channels for evaluation on mismatched conditions. Then, we apply this recipe with a larger training set which contains data from unknown channels and use loss functions that enforce invariance between channels, for the few languages for which unknown channel data is available.



(a) *Task 1* (b) *Task 3*

Figure 1: *Confusion matrices of the submitted system on the evaluation set (number of utterances). Multiple detections are possible. Language labels are defined in Subsection 2.1. Colored numbers correspond to a high number of errors.*

The main difficulty of this challenge is generalization to unknown conditions: new channel for task 1 and noisy conditions for task 3. The analysis of our systems reveals that their performances on our development sets are consistent with results on the evaluation set for unknown channels.The final performance of the submitted system (which achieved best performance for tasks 1 and 3) validates our combination of robust bottleneck features, data augmentation methods and domain robustness loss functions.

10. Acknowledgements

Experiments presented in this paper were carried out using the Grid5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr). This work has been partly funded by the French Direction Générale de l'Armement.

11. 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] D. Snyder, D. Garcia-Romero, A. McCree, G. Sell, D. Povey, and S. Khudanpur, "Spoken language recognition using x-vectors." in *Proc. Odyssey*, 2018, pp. 105–111.

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

[4] M. McLaren, D. Castan, and L. Ferrer, "Analyzing the effect of channel mismatch on the SRI language recognition evaluation 2015 system." in *Proc. Odyssey*, 2016, pp. 188–195.

[5] O. Plchot, P. Matějka, O. Novotný, S. Cumani, A. Lozano-Diez, J. Slavíček, M. Diez, F. Grézl, O. Glembek, M. Kamsali, A. Silnova, L. Burget, L. Ondel, S. Kesiraju, and J. Rohdin, "Analysis of BUT-PT submission for NIST LRE 2017," in *Proc. Odyssey*, 2018, pp. 47–53.

[6] B. M. Abdullah, T. Avgustinova, B. Möbius, and D. Klakow, "Cross-domain adaptation of spoken language identification for related languages: the curious case of slavic languages," in *Proc. Interspeech*, 2020, pp. 477–481.

[7] J. Villalba, N. Brümmer, and N. Dehak, "End-to-end versus embedding neural networks for language recognition in mismatched conditions," in *Proc. ICASSP*. IEEE, 2018, pp. 6199–6203.

[8] F. Castaldo, D. Colibro, E. Dalmasso, P. Laface, and C. Vair, "Compensation of nuisance factors for speaker and language recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 7, pp. 1969–1978, 2007.

[9] Z. Peng, S. Feng, and T. Lee, "Adversarial multi-task deep features and unsupervised back-end adaptation for language recognition," in *Proc. ICASSP*. IEEE, 2019, pp. 5961–5965.

[10] R. Duroselle, D. Jouvet, and I. Illina, "Unsupervised regularization of the embedding extractor for robust language identification," in *Proc. Odyssey*, 2020, pp. 39–46.

[11] D. Wang, L. Li, D. Tang, and Q. Chen, "AP16-OL7: A multilingual database for oriental languages and a language recognition baseline," in *Proc. APSIPA*. IEEE, 2016, pp. 1–5.

[12] Z. Tang, D. Wang, Y. Chen, and Q. Chen, "AP17-OLR challenge: Data, plan, and baseline," in *Proc. APSIPA*. IEEE, 2017, pp. 749–753.

[13] Q. C. Zhiyuan Tang, Dong Wang, "AP18-OLR challenge: Three tasks and their baselines," in *Proc. APSIPA*. IEEE, 2018.

[14] Z. Tang, D. Wang, and L. Song, "AP19-OLR challenge: Three tasks and their baselines," in *Proc. APSIPA*. IEEE, 2019, pp. 1917–1921.

[15] Z. Li, M. Zhao, Q. Hong, L. Li, Z. Tang, D. Wang, L. Song, and C. Yang, "AP20-OLR challenge: Three tasks and their baselines," in *Proc. APSIPA*. IEEE, 2020, pp. 550–555.

[16] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, and R. Pang, "Conformer: Convolution-augmented transformer for speech recognition," in *Proc. Interspeech*, 2020, pp. 5036–5040.

[17] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson, "Averaging weights leads to wider optima and better generalization," in *Proc. Conference on Uncertainty in Artificial Intelligence*, 2018, pp. 876–885.

[18] M. Mclaren, M. K. Nandwana, D. Castán, and L. Ferrer, "Approaches to multi-domain language recognition." in *Proc. Odyssey*, 2018, pp. 90–97.

[19] D. Talkin and W. B. Kleijn, "A robust algorithm for pitch tracking (RAPT)," *Speech coding and synthesis*, vol. 495, p. 518, 1995.

[20] 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 *Proc. Interspeech*, 2002, pp. 89–92.

[21] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proc. ICML*, 2006, pp. 369–376.

[22] Z. Ren, G. Yang, and S. Xu, "Two-stage training for chinese dialect recognition," in *Proc. Interspeech*, 2019, pp. 4050–4054.

[23] S. Ling, J. Salazar, Y. Liu, and K. Kirchhoff, "BERTPHONE: Phonetically-aware encoder representations for utterance-level speaker and language recognition," in *Proc. Odyssey*, 2020, pp. 9–16.

[24] T. Kudo and J. Richardson, "SentencePiece: a simple and language independent subword tokenizer and detokenizer for neural text processing," *CoRR*, vol. abs/1808.06226, 2018.

[25] 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.

[26] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "SpecAugment: A simple data augmentation method for automatic speech recognition," in *Proc. Interspeech*, 2019, pp. 2613–2617.

[27] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *Proc. CVPR*, 2019, pp. 4690–4699.

[28] R. Duroselle, D. Jouvet, and I. Illina, "Metric learning loss functions to reduce domain mismatch in the x-vector space for language recognition," in *Proc. Interspeech*, 2020, pp. 447–451.

[29] N. Brümmer, "The FoCal toolkit," in *website https://sites.google.com/site/nikobrummer/focalmulticlass*, 2007.