ABC System Description for NIST LRE 2022

Anna Silnova[1], *Josef Slavicek*[2], *Ladislav Mošner*[1], *Michal Klčo*[2], *Oldřich Plchot*[1], *Pavel Matějka*[1,2], *Junyi Peng*[1], *Themos Stafylakis*[3] *Lukáš Burget*[1],

(1) BUT, Speech@FIT, Czech Republic
(2) Phonexia, Czech Republic
(3) Omilia, Greece
{matejkap|iplchot|isilnova}@vut.cz[1],
josef.slavicek@phonexia.com[2], tstafylakis@omilia.com[3]

Abstract

1. Introduction

Our submission is a collaborative effort of BUT, Phonexia and Omilia.

2. Data

2.1. Fixed condition

For the fixed training condition we used only data allowed for this condition by the evaluation plan [1]:

- 2017 NIST LRE Development Set and previous NIST LRE training data (LDC2022E16)
- 2017 NIST LRE Test Set (LDC2022E17)
- 2022 NIST LRE Development Set (LDC2022E14)
- VoxLingua107 [2]

2.2. Data for Open condition

There are 2 systems trained for open condition and each is trained on different datasets, therefore it will be described later within the description of the systems.

2.3. BUTdev

This data is a dev set for NIST LRE 2022. It was used to train one backend for open condition system. It contains 14 target languages from NIST LRE 2022 but is different from LDC2022E14. It has 8491 files/trials, 34 hours of speech after energy based VAD. For most of the languages we had between 20 to 120 minutes of data, only for 2 languages we had more (4 and 18 hours). The data come from LWAZI [3], ADI17 [4], LORELEI [5], NIST LRE 2015, NIST SRE 2019 and VOA[1].

The work was supported by Czech Ministry of Interior project No. VJ01010108 "ROZKAZ", Czech National Science Foundation (GACR) project NEUREM3 No. 19-26934X, Czech Ministry of Education, Youth and Sports project no. LTAIN19087 "Multi-linguality in speech technologies", and Horizon 2020 Marie Sklodowska-Curie grant ESPERANTO, No. 101007666. Computing on IT4I supercomputer was supported by the Czech Ministry of Education, Youth and Sports through the e-INFRA CZ (ID:90140).

[1]https://www.voanews.com/

3. Subsystems

3.1. VAD

If not stated otherwise we used energy based VAD for systems described below. For each test file we train Gaussian Mixture model (GMM) in unsupervised fashion on this recording. The GMM have 3 Gaussian components. We discard the frames assigned to the Gaussian which represents the lowest energy frames in the recording. This VAD does not use any pretrained parameters.

3.2. Fixed condition – ResNet100

The ResNet100 system was inspired by the speaker ID system from [6] with a different number of output channels for each stage (32, 64, 128, 256). In each ResNet block, the frequency-wise Sqeeze-Excitation (fwSE) [7] block was incorporated with a bottleneck size of 128.

The input features were 80-dimensional log Mel-filterbanks extracted with a window length of 25 ms and a frame-shift of 10 ms. On randomly selected 4s chunks of training utterances, we applied Kaldi-style augmentation using MUSAN [8] corpus and RIR [9] corpus (additive noise, music, and reverberation). The features were mean-normalized with a floating window of 3s.

The system was trained for 3 epochs on training examples generated by KALDI (egs archives) with AM-softmax loss, where we set the margin to 0 and the scale to 32. The learning rate was scheduled in the same way as in [6]. For the 6% of the training process (warmup), the learning rate was linearly increased from 1e-5 to 0.1, then, the learning rate was fixed for 20% of the training process (plateau) and exponentially decreased with a rate of 0.5 after each 8% of the training process for the rest of the training. Stochastic Gradient Descent (SGD) was used with a weight decay of 1e-4 and momentum of 0.9. The network was trained on 4 GPUs with a total batch size of 512 (128 per GPU) and syncbatchnorm per 2 GPUs.

The system is 160x FTRT (excluding feature and VAD calculation) on RTX 2080 Ti GPU, and its memory consumption is roughly 900MB.

3.3. Fixed condition – ResNet34-CE, ResNet34-AAM, ResNet34-2head

Models ResNet34-CE, ResNet34-AAM, and ResNet34-2head are based on the same backbone architecture. They differ in slight modifications of the training procedure. What follows is the description of shared settings across the models.

The backbone architecture is derived from the standard vision model [10]. As opposed to the original ResNet34, the first convolution has a kernel size of 3 and a stride of 1. The stages of the network have (64, 128, 256, 256) channels. 64-dimensional Mel-filterbank features constitute the input to the model, where window length and shift are 25 ms and 10 ms, respectively. We limit the input example length to 300 frames. After the initial frame-wise processing, internal features are aggregated with statistics (mean and standard deviation) pooling. Statistics are projected to 256-dimensional embeddings subsequently utilized in the backend stage. The models were trained for language classification on 8-GPU compute nodes by SGD with a momentum of 0.9. We regularized the models with a weight decay with a factor of 1e-4. Each per-replica minibatch comprised 64 examples. During the initial warm-up stage (first 10k training steps), the learning rate increased from 0 to 0.2. Subsequently, it was multiplied by a factor of 0.5 every time a plateau on a cross-validation loss was reached. The cross-validation set was derived from the training set.

The models differ in the objective function they minimized. ResNet34-CE was trained with a cross-entropy objective, whereas for ResNet34-AAM, we used additive angular margin loss (AAM) [11] with a margin set to 0 and a scale to 30. Both ResNet34-CE and ResNet34-AAM were trained on the VoxLingua107 dataset [2]. We did not apply VAD for VoxLingua107 corpus. Development and evaluation data of the challenge contain fine-grained labels (at the level of dialects), which contradicts the coarse-grained labels of VoxLingua107 (e.g., for English and Arabic language). To allow the ResNet34-AAM network to learn more than one representation per VoxLingua107 languages, we adopted sub-center ArcFace [12] with 3 sub-centers.

ResNet34-2head optimized the same AAM objective as ResNet34-AAM and also utilized 3 sub-centers for classes. Contrary to previous systems, it was trained on VoxLingua107 and NIST LRE 2017 datasets. They contain labels of different granularity. To tackle this issue, the model has two classification heads (one for each corpus). Head and example correspondence to the dataset is taken into account when computing the loss.

The ResNet34 models are approximately 52x FTRT (including feature extraction and VAD calculation) on Nvidia A100-SXM4 GPU, and their memory consumption is around 640MB.

3.4. ECAPA-TDNN

As an alternative to ResNet systems, we trained also ECAPA-TDNN [13] where we followed the recipe from SpeechBrain[2]. The model contains approximately 14M parameters and is trained with AAM-Softmax with a margin set to 0. The targets are the languages contained in the LDC2022E16 and Voxlingua datasets. All data were downsampled to 8KHz before training.

The ECAPA-TDNN model runs approximately 70x FTRT (including feature extraction and VAD calculation) on Nvidia A100-SXM4 GPU, and the memory consumption is around 400MB.

[2]`https://github.com/speechbrain/speechbrain/`

3.5. Fixed condition – RepVGG

The system based on the RepVGG architecture utilizes the B1 variant from [14]. The training setup and parameters are the same as in the training of ResNet100 except for batch size, which is 380 in total (95 per GPU).

The RepVGG-based system is 100x FTRT (excluding feature and VAD calculation) on RTX 2080 Ti GPU, and its memory consumption is 1170MB.

3.6. Open condition – ResNet101

The ResNet101 model is, in essence, a scaled-up version of ResNet34 networks. Therefore, it shares multiple settings, including the number of channels per stages (64, 128, 256, 256), statistics pooling, embeddings dimensionality, and optimization along with learning rate scheduling. Due to increased GPU memory requirements, each per-replica minibatch contained 32 examples. Input feature extraction is equivalent to that for ResNet34 models. The model was trained with the AAM objective with a margin of 0 and a scale of 30.

The ResNet101 model is approximately 46x FTRT (including feature extraction and VAD calculation) on Nvidia A100-SXM4 GPU, and its memory consumption is around 840MB.

3.6.1. Training data

The system was trained on two datasets — VoxLingua107 and an in-house one. Our dataset comprises 106 languages, 730k utterances, 9730 hours of speech after energy based VAD. It is colection of these datasets:

- LDC data - CallFriend, CallHome, Foreign accented english, Fisher, HKUST Mandarin, NIST LRE & SRE data, VOA, OGI, OGI22
- data from projects - BABEL, LORELEI [5], MATERIAL, WELCOME
- other sources - KALAKA [15], ADI17 [4], LWAZI [3], SpeechDat(E) from ELRA, MSIL18 [16], Radio Free Europe [3]

3.7. Open condition – XLSR Models

For open condition, we used two systems based on pretrained `xls-r-1b` [17], downloaded from HuggingFace[4]. We finetune the systems as language classifiers with TDNN classification head. We adopted TDNN architecture from [18]. The systems are finetuned with crossentropy loss with Adam [19] optimizer. The only difference between our two `xls-r` systems is the learning rate.

3.7.1. Training data

We finetuned our `xls-r` systems on following data: VoxLingua107[5][2], LWAZI[6], and telephony dataset internally called PHXManylang, containing 7300 h of speech (after processing with VAD) in 325 000 utterances and 83 languages. Main ingredients of PHXManylang are: CommonVoice (eu, cy, es, sl, nl, et, fa, ca, de), LDC datasets 96S46, 96S47, 96S48, 96S49, 96S50, 96S51, 96S52, 96S53, 96S54, 96S55, 96S56, 96S57, 96S58, 96S59, 96S60, 2004S13, 2005S07, 2005S15, 2006S29, 2006S34, 2006S43, 2006S44, 2007S02, 2009S05, 2011S01, 2011S04,

[3]https://www.rferl.org/

[4]https://huggingface.co/facebook/wav2vec2-xls-r-1b

[5]http://bark.phon.ioc.ee/voxlingua107/

[6]https://sites.google.com/site/lwazispeechcorpus?pli=1

2011S05, 2011S07, 2011S09, 2011S10, 2013S02, 2013S04, 2013S07, 2013S08, 2014S06, 2015S07, 2016S02, 2016S06, 2016S08, 2016S09, 2016S10, 2016S12, 2016S13, 2017S01, 2017S03, 2017S05, 2017S08, 2017S13, 2017S19, 2017S22, 2018S02, 2018S07, 2018S13, 2018S16, 2019S03, 2019S08. Another part of PHXManylang is telephony speech extracted from VOA broadcast recordings – 2200 hours, 51 000 utterances, 42 languages (out of which Ndebele, Oromo and Tigrinya are among NIST LRE22 target languages). Last part of PHXManylang are proprietary datasets, mostly from callcenters – 1100 hours, 87 000 utterances, 13 languages (hr, cs, nl, en, fa, fr, de, it, lb, pl, ru, sk, es).

Data from VoxLingua107 were used without augmentation and without VAD. For data from other datasets, we used Kaldi [20] augmentation (reverberation, Musan noise, Musan music), and we applied VAD. Our VAD implementation is based on BUT Hungarian phoneme recognizer[7], which vas converted to VAD by mapping all phonemes to speech.

3.7.2. *Training procedure*

We construct training batches by iterating over all utterances in our dataset and choosing randomly 4s chunks from each utterance. Most of our data is 8 kHz, so we upsample it to 16 kHz, which is input expected by `xls-r`. For VoxLingua107, which is 16 kHz, we randomly, with 50% probability, downsample chunks to 8 kHz and upsample back to 16 kHz.

We ran our finetuning in `bloat16` on 2 GPUs with batch size 32 in total (i.e., 16 per GPU), without syncbatchnorm. Training script is implemented in Torch + HuggingFace and we switch off some features which are the default for HuggingFace `xls-r` in train mode: layerdrop and random input masking. Similarly, we are keeping `xls-r` dropouts in eval mode during training. We apply weight decay 1.0e-6.

We finetune for 6 epochs. For TDNN head, we are using the same learning rate for both our systems, with a scheduler which first warms up the learning rate to the nominal value, then keeps it constant, and then decays it back to zero. Warm up takes 0.05 T and the constant phase 0.475 T, where T is the duration of the whole training. Our nominal learning rate for the TDNN head is 1.0e-3. For `xls-r` backbone, we first keep it frozen (LR=0) for 0.3 T, then we warmup LR for 0.1 T and then decay (without constant phase). The nominal `xls-r` learning rate for our systems is 1.0e-6 and 3.0e-6, respectively.

3.7.3. *Embedding extraction*

For embedding extraction, we upsample the competition data to 16kHz and process them with a sliding window of maximal length 20s. We scan the whole utterance by shifting the window by 2s and then average embeddings obtained from all windows. This approach is very inefficient because it scans the same part of audio up to 10x. So it's only 8x FTRT on RTX 2080 Ti GPU, with 9GB memory consumption.

4. Classifiers

For the majority of the embeddings, the final log-likelihood scores were generated by Gaussian Linear Classified (GLC). GLC is the generative model that assumes that for each class, the embeddings $\mathbf{r}_j$ were sampled from the Gaussian distribution:

$$\mathbf{r}_j \sim \mathcal{N}(\mathbf{m}_i, \mathbf{\Lambda}^{-1}), \tag{1}$$

[7]https://speech.fit.vutbr.cz/software/phoneme-recognizer-based-long-temporal-context

where $\mathbf{m}_i$ is the class-dependent mean vector, and $\mathbf{\Lambda}^{-1}$ is a covariance matrix shared by all classes. Then, the class-conditional log-likelihood for $\mathbf{r}_j$ given language i can be obtained as:

$$\log P(\mathbf{r}_{ij} \mid i) = \frac{1}{2}\log|\mathbf{\Lambda}| - \frac{1}{2}(\mathbf{r}_j - \mathbf{m}_i)^T \mathbf{\Lambda}(\mathbf{r}_j - \mathbf{m}_i) + k, \tag{2}$$

where k is the constant, not depending on the data. The parameters of GLC are obtained by Maximum-Likelihood estimation. In all cases, we use half of the LRE22 development set to estimate the GLC parameters[8]. For ResNet34 systems from the fixed condition and ResNet101 from the open condition, the training data were expanded by including the embeddings extracted from the augmented version of LRE22 development set (one augmented utterance per one original). Finally, for ResNet101 embeddings, we trained the second GLC backend on the augmented version of LRE22 development set and BUTdev utterances.

Unlike all the other systems, the embeddings extracted from XLS-R-1b (3e-6) model were classified using Probabilistic Linear Discriminant Analysis (PLDA) model. In this case, we train the PLDA on a half of LRE22 development set using languages as labels. Then, to generate the vector of scores, each of the test utterances is scored against 14 enrollment models. Each enrollment model is composed of training utterances belonging to the same language. We average the embeddings from the enrollment segments sharing the same session id, i.e., the final score for each language is the log-likelihood ratio (LLR) score for the multi-enroll/single-test verification trial, with the number of enrollment segments equal to the number of sessions used from a given language.

For most of the systems, before training the back-end (GLC or PLDA), we reduce the dimensionality of the embeddings by Principal Component Analysis (PCA); the respective target PCA dimensionalities are reported in Table 1. For some of the embeddings, the target dimensionality after PCA was selected automatically: there was an obvious gap in the eigenvalues of the covariance matrix estimated on the training data, i.e., some dimensions had very little variability compared to the others. For the embeddings, where we did not observe a similar pattern: the eigenvalues were gradually increasing when sorted, we experimented with several options for PCA dimensionality and selected the one performing the best on the held-out development set.

5. Calibration and fusion

After retrieving the scores from the individual systems, we proceeded with two additional steps: calibration and fusion.

We utilize Logistic Regression (LR) calibration: a single scalar and a vector of offsets (one per language) are learned by optimizing the cross-entropy objective. We used a uniform prior over 14 language classes when computing the objective. During model development, we trained the calibration parameters on a half of the LRE22 development set and tested the performance on the held-out half of the development set. The calibration parameters are learned on the same data as was used to train the

[8]Notice that we used only one half of LRE22 development set for training the backend and calibration parameters during the system development. The second half was used to track the performance. This is consistent with the description of the classifiers and calibration and with the results of Table 1. For the final submission, the selected systems were retrained on the whole LRE22 development set and submitted without testing.

classifier. When training the final model for the submission, we used the whole LRE22 development set for both back-end and calibration training.

The calibration stage was applied to all of the systems submitted to the fixed condition and to two systems eligible for the open condition: SBN i-vector and one of ResNet101. The remaining three systems submitted to the open condition were not calibrated due to the low amount of errors that they had on the calibration set.

In all cases, the fusion step did not require training any parameters and consisted of averaging the (calibrated) scores from the individual systems.

6. References

[1] Yooyoung Lee, Craig Greenberg, Lisa Mason, and Elliot Singer, "The 2022 NIST language recognition evaluation plan (lre22)," https://www.nist.gov/publications/nist-2022-language-recognition-evaluation-plan.

[2] Jörgen Valk and Tanel Alumäe, "Voxlingua107: A dataset for spoken language recognition," *2021 IEEE Spoken Language Technology Workshop (SLT)*, pp. 652–658, 2020.

[3] Charl van Heerden adn Neil Kleynhans and Marelie Davel, "Improving the lwazi asr baseline," in *Interspeech*, 2016.

[4] Suwon Shon, , Ahmed Ali, Younes Samih, Hamdy Mubarak, and James Glass, "Adi17: A fine-grained arabic dialect identification dataset," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 8244–8248.

[5] Stephanie Strassel and Jennifer Tracey, "Lorelei language packs: Data, tools, and resources for technology development in low resource languages," in *International Conference on Language Resources and Evaluation*, 2016.

[6] Rostislav Makarov, Nikita Torgashov, Alexander Alenin, Ivan Yakovlev, and Anton Okhotnikov, "Id r&d system description to voxceleb speaker recognition challenge 2022," *ID R&D Inc.: New York, NY, USA*, 2022.

[7] Jenthe Thienpondt, Brecht Desplanques, and Kris Demuynck, "Integrating frequency translational invariance in tdnns and frequency positional information in 2d resnets to enhance speaker verification," *arXiv preprint arXiv:2104.02370*, 2021.

[8] David Snyder, Guoguo Chen, and Daniel Povey, "Musan: A music, speech, and noise corpus," *arXiv preprint arXiv:1510.08484*, 2015.

[9] Tom Ko, Vijayaditya Peddinti, Daniel Povey, Michael L Seltzer, and Sanjeev Khudanpur, "A study on data augmentation of reverberant speech for robust speech recognition," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 5220–5224.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[11] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou, "ArcFace: Additive Angular Margin Loss for Deep Face Recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[12] Jiankang Deng, Jia Guo, Tongliang Liu, Mingming Gong, and Stefanos Zafeiriou, "Sub-center ArcFace: Boosting Face Recognition by Large-Scale Noisy Web Faces," in *Computer Vision – ECCV 2020*, 2020, pp. 741–757.

[13] Brecht Desplanques, Jenthe Thienpondt, and Kris Demuynck, "ECAPA-TDNN: Emphasized Channel Attention, Propagation and Aggregation in TDNN Based Speaker Verification," in *Interspeech2020*, 2020, pp. 1–5.

[14] Miao Zhao, Yufeng Ma, Min Liu, and Minqiang Xu, "The speakin system for voxceleb speaker recognition challange 2021," *arXiv preprint arXiv:2109.01989*, 2021.

[15] Luis Javier Rodríguez-Fuentes, Mikel Penagarikano, Germán Bordel, Amparo Varona, and Mireia Díez, "KALAKA: A TV broadcast speech database for the evaluation of language recognition systems," in *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, May 2010, European Language Resources Association (ELRA).

[16] Brij Mohan Lal Srivastava, Sunayana Sitaram, Rupesh Kumar Mehta, Krishna Doss Mohan, Pallavi Matani, Sandeepkumar Satpal, Kalika Bali, Radhakrishnan Srikanth, and Niranjan Nayak, "Interspeech 2018 low resource automatic speech recognition challenge for indian languages," in *Workshop on Spoken Language Technologies for Under-resourced Languages*, 2018.

[17] Arun Babu, Changhan Wang, Andros Tjandra, Kushal Lakhotia, Qiantong Xu, Naman Goyal, Kritika Singh, Patrick von Platen, Yatharth Saraf, Juan Pino, Alexei Baevski, Alexis Conneau, and Michael Auli, "XLS-R: self-supervised cross-lingual speech representation learning at scale," *CoRR*, vol. abs/2111.09296, 2021.

[18] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust dnn embeddings for speaker recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018.

[19] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[20] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al., "The kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*. IEEE Signal Processing Society, 2011, number CONF.

[21] Pavel MATĚJKA et. al., "But- pt system description for nist lre 2017," in *National Institute of Standard and Technology*, 2017.

Table 1: *The performance of the submitted systems on the half of LRE 2022 development set. The second half was used to train the classifier and calibration parameters. The systems marked with "*" are submitted as contrastive single systems.*

	Embeddings	Back-end	Calibration	minC	actC	minC	actC
		Fixed condition					
1*	ResNet100	PCA 117, GLC	LR	0.294	0.314	0.250	0.256
2	RepVGG	PCA 117, GLC	LR	0.340	0.373	0.266	0.275
3	ResNet34-CE	PCA 100, GLC	LR	0.330	0.354	0.268	0.275
4	ResNet34-2head	GLC	LR	0.274	0.296	0.228	0.239
5	ResNet34-AAM	PCA 100, GLC	LR	0.313	0.329	0.283	0.284
6	ECAPA-TDNN	PCA 100, GLC	LR	0.409	0.458	0.342	0.354
		Primary submission 1+2+3+4		0.235	0.243	0.191	0.193
		Contrastive submission 1+2+3+4+5+6		0.236	0.244	0.193	0.194
		Open condition					
7*	XLS-R-1b LR=1e-6	PCA 100, GLC	-	0.113	0.123	0.102	0.110
8	XLS-R-1b LR=3e-6	PCA 300, PLDA	-	0.114	0.130	0.095	0.104
9	ResNet101	PCA 151, GLC	-	0.167	0.182	0.152	0.163
10*	ResNet101	PCA 151, GLC(+BUTdev)	LR	0.172	0.181	0.148	0.152
11*	LRE 2017 BUT BabelSBN ivectors [21]	PCA 100, GLC	LR	0.379	0.392	0.307	0.310
		Primary submission 7+8+9		0.097	0.108	0.082	0.088