ABC NIST SRE 2019 CTS SYSTEM DESCRIPTION

Jahangir Alam[5], *Gilles Boulianne*[5], *Ondřej Glembek*[1], *Alicia Lozano-Diez*[1,6], *Pavel Matějka*[1,2], *Petr Mizera*[4], *Joao Monteiro*[5], *Ladislav Mošner*[1], *Ondřej Novotný*[1], *Oldřich Plchot*[1], *Johan Rohdin*[1], *Anna Silnova*[1], *Josef Slavíček*[2], *Themos Stafylakis*[4], *Shuai Wang*[1,3], *Hossein Zeinali*[1]

[1]Brno University of Technology, Speech@FIT and IT4I Center of Excellence, Brno, Czechia
{matejkap,iplchot,...}@fit.vutbr.cz
[2]Phonexia, Czechia
slavicek@phonexia.com
[3]Speechlab, Shanghai Jiao Tong University, China
feixiang121976@sjtu.edu.cn
[4]Omilia - Conversational Intelligence, Athens, Greece
tstafylakis@omilia.com
[5]CRIM, Montreal (Quebec), Canada
jahangir.alam@crim.ca
[6]Audias-UAM, Universidad Autonoma de Madrid, Madrid, Spain
alicia.lozano@uam.es

1. INTRODUCTION

This submission is a collaborative/competitive effort of BUT, Phonexia, Omilia, CRIM and UAM. All systems in fusions are based on x-vector paradigm with different features, DNN topologies and backends.

2. TELEPHONE SYSTEMS - CMN2

2.1. Training data, Augmentation

For training the networks, we used the following:

- SRE 4, 5, 6, 8, 10
- Fisher Arabic
- All switchboard data
- Voxceleb 1 and 2

For performed the following data augmentations which are the same as in Kaldi recipe except for the compression:

- Reverberated
- Augment with Musan noise
- Augment with Musan music
- Augment with Musan speech
- Compression using ogg and mp3 codecs

After creating a list of utterances for augmentation, a subset of 500K utterances from this list was selected and added to the training data. Afterwards, utterances with less than 500 frames and also speakers with less than 5 training utterance were removed. Finally, the training data for creating training archives contained 17054 speakers.

2.1.1. VAD, Input features

We used FBANK features and energy-based VAD from Kaldi SRE16 recipe without any modification. The features are therefore 40-dimensional FBANKs which are extracted from 25 ms windows with 15 ms overlap. The bandwidth is limited between 20 and 3700 Hz.

2.1.2. Training archives

For creating the training archives, we used Kaldi-like archive generation for our Tensorflow implementation and therefore, for both Kaldi and Tensorflow, the same configuration was used for generating two different sets of archives. Minimum and maximum number of frames in each training example are 200 and 400 respectively. Number of repeats for each speaker is 25 and maximum number of frames per archive is 2 billion. By using this configuration, two sets of 124 archives were generated for Kaldi and Tensorflow versions.

Authors are in alphabetical order.

The work was supported by Czech Ministry of Interior project No. VI20152020025 "DRAPAK", Google Faculty Research Award program, Czech Science Foundation under project No. GJ17-23870Y, and by Czech Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project "IT4Innovations excellence in science - LQ1602". It was also supported by Technology Agency of the Czech Republic project No. TJ01000208 "NOSICI", European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No. 748097, the Marie Sklodowska-Curie cofinanced by the South Moravian Region under grant agreement No. 665860, and by the U.S. DARPA LORELEI contract No. HR0011-15-C-0115. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

Table 1. ResNet50 architecture, N in the last row is the number of speakers. The first dimension of the input shows number of filterbanks and the second dimension indicates the number of frames.

Layer name	Structure	Output
Input	–	$40 \times 200 \times 1$
Conv2D-1	3×3, Stride 1	$40 \times 200 \times 32$
ResBlock-1	$\begin{bmatrix} 1 \times 1, 32 \\ 3 \times 3, 32 \\ 1 \times 1, 128 \end{bmatrix} \times 3$, Stride 1	$40 \times 200 \times 128$
ResBlock-2	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 4$, Stride 2	$20 \times 100 \times 256$
ResBlock-3	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 6$, Stride 2	$10 \times 50 \times 512$
ResBlock-4	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 3$, Stride 2	$5 \times 25 \times 1024$
StatsPooling	–	10×1024
Flatten	–	10240
Dense1	–	256
Dense2 (Softmax)	–	N
Total	–	–

2.2. ResNet

ResNet [1] based embeddings are extracted from a standard 50-layer ResNet (ResNet50). This network uses 2-dimensional features as input and processes them using 2-dimensional CNN layers. Inspired by x-vector topology, both mean and standard deviation are used as statistics. The detailed topology of the used ResNet is shown in Table 1. The ResNet was trained using SGD optimizer for 6 epochs.

2.3. F-TDNN

For this system we use the factorized TDNN architecture proposed in [2]. We train it with the Kaldi toolkit [3] with the settings in the `sre16/v2` recipe except that we used the data set described in Sections 2.1.1 and 2.1.2 and that we trained the model for six epochs instead of three.

2.4. Res-E-TDNN with Denoising

For this system, we use a modified version of E-TDNN architecture [2]. In each frame-level layer of the network 768 outputs (channels) are used instead of 512. Also, based on our experience for VoxCeleb challenge [4], we add few residual connections in the frame-level layers. The input of each liner layer in the frame-level part is a summation of the output of all previous TDNN layers. So, the first linear layer receives the input from one TDNN layer, the second one receives it from summation of two TDNN layers and so on. We train it with the Kaldi toolkit and data described in Section 2.3 and we trained the model for three epochs. During extraction, each audio file was pre-processed with denoising based on neural-network autoencoder. The autoencoder, training data, and augmentation are described in [5] and [6].

2.5. Res-E-TDNN with adversarial adaptation

In this approach we begin with a Res-E-TDNN network (see sect. 2.4) pretrained on SRE English telephone data and we apply adversarial domain adaptation. Similarly to our approach described in [7], we attach a domain discriminator (feed-forward neural network with 3 hidden layers and Leaky ReLU) which aims at discriminating between source and target domains (English and Arabic, respectively). The x-vector extractor tries to fool the discriminator by maximizing the binary cross entropy loss of the discriminator. The adversarial loss encourages the extractor to encode utterances into x-vectors that are hard to distinguish in terms of domain (i.e. language). Moreover, as the divergence between the two marginal distributions becomes smaller, a PLDA model trained solely on source x-vectors can perform fairly well on the target domain without any adaptation. Note that different from [7], we use a standard GAN (as opposed to Wasserstein GAN) and we do not augment input features or hidden representations with domain labels. During finetuning, the number of source training speakers is 4254 selected from SRE 2004-2010 (English telephone only), while we augment the target domain data with Fisher Arabic resulting in 2251 Arabic speakers. Two different softmax layer are used (one for each domain) and their associated cross entropy losses are added, yielding the overall speaker classification loss.

2.6. Backend

2.6.1. Training data

All of the backends utilized the following data.

- Training set - data from Mixer collection (NIST SRE 2004-2010) from which we kept only telephone recordings, approximately 66k utterances.
- Adaptation set - SRE18 development set and 60% of the data from SRE18 evaluation dataset. The resulting set consisted of 8k utterances coming from 137 speakers.
- Snorm data - part of the adaptation set (5 utterances per speaker) and SRE18 unlabeled data.

2.6.2. General pipeline

Here, we describe our general strategy for the backend training. And, in the following sections, we discuss what modifications were made for each particular subsystem. One of the systems from the submitted fusion used Gaussian-PLDA (GPLDA) backend, the other three utilized heavy-tailed-PLDA (HT-PLDA)[8].

For all of the systems, we start with the mean normalization. Evaluation data were centered using the mean computed on adaptation set, while the training data are centered using their own mean. Then, we apply feature-distribution adaption (FDA) transformation [9] for the training data. The goal of the transformation is to modify the out-of-domain training data so that their covariance is not lower than the covariance of the in-domain adaptation data in any direction. Then, Gaussian or heavy-tailed PLDA model is trained using the transformed training set. And, additionally, we train "adaptation" model on the untransformed adaptation set. The final adapted model was derived from the two PLDA models so that the modeled across-speaker covariance matrix is an average of the covariance matrices from the constituent models. Similarly, the parameters describing within-speaker covariance matrix are also interpolated. Finally, we applied adaptive score normalization using top 800 scoring files from the snorm set.

2.6.3. *ResNet_GPLDA*

When training this system, we added SRE16 evaluation data (10k utterances from 201 speakers) to the training set. Additionally to the centering and FDA preprocessing mentioned before, we applied LDA, reducing the dimensionality of embeddings from 256 to 250, and length normalization. Then, we trained GPLDA model for which the size of speaker and channel subspaces was set to 150. Adaptation model is also GPDA, but the speaker and channel subspaces are smaller. Their size was set to 50.

2.6.4. *FTDNN_HTPLDA*

For FTDNN embedding system, we trained HTPLDA model with the size of the speaker subspace set to 200 on the length-normalized embeddings from the training set enlarged by adding 4 types of the augmentations for each training utterance. Then, a smaller HTPLDA model (speaker space of size 100) is trained on the adaptation data. Degrees of freedom parameter for both models was set to 2.

2.6.5. *Res-E-TDNN_DENOISE_HTPLDA*

In this case, we trained HTPLDA model with the size of the speaker subspace set to 200 on the 66k embeddings from the training set. SRE16 evaluation data were added to the adaptation set. Adaptation HTPLDA of the same size as the main one was trained on the resulting data. Degrees of freedom parameter for both models was set to 2.

2.6.6. *Res-E-TDNN_GAN_HTPLDA*

The backend training of this model practically repeats what we did for the FTDNN. The only difference is that we don't use length normalization in this case. All of the other parameters are the same as above.

3. CALIBRATION & FUSION

The final submission strategy was one common fusion trained on the labeled development set created by holding out 40% of the NIST SRE2018 CMN2 evaluation data. Each system provided log-likelihood ratio scores that could be subjected to score normalization. These scores were first pre-calibrated and then passed into the fusion. The output of the fusion was then again re-calibrated.

Both calibration and fusion was trained with logistic regression optimizing the cross-entropy between the hypothesized and true labels on a corresponding development set. Our objective was to improve the error rate on the development set. We observed very similar error rates on our development set and NIST's progress set during submitting our intermediate systems to the NIST leader-board.

4. SUBMISSION & RESULTS

Table 2 shows the performance of the individual systems on our internal development set as well as the performance of the final fusion on our dev set along with the result on the progress set from the leader-board.

5. CPU USAGE FOR SINGLE X-VECTOR SYSTEM

In single threaded setup on Intel(R) Xeon(R) CPU E5-2670 0 @ 2.60GHz, the x-vector extraction time is of 8.0 times faster than real time (FRT) (computed only on detected speech, would be 12.6 FRT computed for whole recordings including silence). Memory consumption is 500 MB for typical utterance. Other computation (PLDA, cosine distance, calibration, fusion) is negligible.

6. SYSTEMS DEVELOPED AT CRIM

For NIST-SRE 2019 Conversational Telephone Speech (CTS) task, we developed speaker verification systems following two different strategies: (i) we follow the well known x-vector setting [10] in its Kaldi [3] implementation and improve on top of that by evaluating several data augmentation schemes. (ii) Our own Pytorch [11] implementation of variations of x-vectors TDNNs and ResNets, on top of which we introduce several modifications such as a multi-task setting in which speaker recognition metric learning are performed jointly to enforce discriminable features, online hard triplets selection, entropy regularization, learning rate warm-up, and label smoothing. We further evaluate a TDNN augmented with a pyramidal pooling layer [12] over the time dimension as an alternative to the simple statistics pooling layer in [10].

6.1. Multi-class classification based systems

Speaker recognition, i.e. multi-class classification over the set of training speakers, has been successfully applied as an auxiliary task for automatic speaker verification (ASV). Outputs of some inner layer of the model trained under that setting can be then used for PLDA training and inference, of for direct scoring using cosine similarity, for instance. To this end, we adopted extended TDNN and factored TDNN (F-TDNN) based x-vector extraction paradigm. As backend, we employed PLDA.

In this case, In order to adapt x-vectors of the out-of-domain data (i.e., PLDA training data) to the in-domain data (i.e., SRE 2019 or SRE 2018 domain) unsupervised domain adaptation by correlation alignment [13] has been applied. This adaptation technique works by aligning the distributions of out-of-domain and in-domain features in an unsupervised way. This is achieved by aligning second-order statistics, i.e covariances.

6.1.1. *Training data*

Two training data set were used for training TDNN and F-TDNN models - (1) Data corresponding to SRE's from 04 to 10, Mixer 6, and Switchboard from approximately 5000 speaker and (2) combined voxceleb 1 & 2 (excluding the voxceleb1 test set) corpora, which sums up to approximately 7300 speakers.

6.2. Multi-task systems

We closely follow the setting introduced for ResNets in [14] which means to augment the speaker recognition setting described above with triplet loss minimization, performed jointly during training with the goal of enjoying the benefits of the two approaches, i.e.: (i) the relative easiness of training under the multi-class classification setting when compared to metric learning alone, and (ii) the discriminability provided by the triplet loss minimization resulting representations. Triplet loss will be computed on top of embeddings projected on the unit sphere: $y^p = \frac{y}{||y||_2}$, and for the speaker recognition term, a softmax output layer is employed so that the multi-class cross entropy can be computed using speaker identities as class labels. Training is performed so as to minimize the sum of the two losses. Three architectures are evaluated in this case:

Table 2. Results of the single systems on our SRE19 dev set

#	System	ABC SRE19 dev			SRE19 Leaderboard		
		minDCF	actDCF	EER (%)	minDCF	actDCF	EER (%)
1	ResNet_GPLDA	0.281	0.285	3.77	-	-	-
2	FTDNN_HTPLDA	0.255	0.257	3.37	-	-	-
3	Res-E-TDNN_DENOISE_HTPLDA	0.312	0.314	4.30	-	-	-
4	Res-E-TDNN_GAN_HTPLDA	0.273	0.275	3.76	-	-	-
	1+2+3+4	0.216	0.218	3.00	0.216	0.220	2.9

- modTDNN - A variation of the original x-vector TDNN without dilations in the convolutional layers.
- ASPP - A TDNN augmented with an pyramidal pooling layer [12] right before statistics pooling. Such pooling layer consists in summing up the outputs of for convolutional+max. pool layers with different kernel sizes.
- A very deep ResNet-101 modified so as to operate over the time dimension only.

6.2.1. Training data

In the case of Multitask systems, training is performed in two steps. We first train our models on features extracted from the data corresponding to SRE's from 04 to 10, Mixer 6, Switchboard, and Fisher corpora, which sums up to approximately 20000 speakers. Our best systems are thus fine tuned for the same number of iterations on the development data comprised of SREs 12, 16, and 18 (all dev and a small portion of eval data).

6.3. Speech features, VAD and data augmentation

Speech features correspond to 23 MFCCs (Mel-frequency Cepstral Coefficients) obtained with a short-time Fourier transform using a 25ms Hamming window with 10 ms frame shift. For voxceleb, data is down-sampled to 8kHz. An energy-based voice activity detector is employed to filter out non-speech frames. Multi-condition training data is further introduced by augmenting the original train partition with supplementary noisy speech in order to enforce model's robustness across varying conditions. We thus created additional versions of training recordings as similarly done in [10], i.e. by corrupting original samples adding reverberation (reverberation time varies from 0.25s - 0.75s), as well as by adding background noise such as music (signal-to-noise ratio, SNR, within 5-15dB), and babble (SNR varies from 10 to 20dB). Noise signals were selected from the MUSAN corpus [15] and the room impulse responses to simulate reverberation from [16]. We have also developed one system employing TDNN on the top of 23-dimensional perceptual linear prediction (PLP) features.

6.4. Back-end

PLDA was employed for scoring trials after dimensionality reduction of embeddings using linear discriminant analysis (LDA). PLDA is trained on embeddings from the train partition with the same augmentation used for training the neural networks. The model adaptation scheme introduced in [17] was further evaluated and utilized for PLDA to help on overcoming any domain shift observed across train and evaluation data due to different recording conditions and language mismatch. To do so, embeddings unlabelled data are then employed for training a second PLDA model. The final back-end is obtained by simply averaging the covariance matrices of the two PLDA models. We found the adaptation described to have different impact in performance depending on the underlying model utilized for generating the embeddings, and in some cases some performance degradation was observed. We further highlight that reported results are obtained from models that achieved the minimal validation loss throughout training, and the evaluation data is only made available to the models to generate the reported metrics, not being used at development phase.

Table 3. Results of the single systems on our SRE19 dev set

System	minDCF	actDCF	EER (%)
ASPP (MFCC)	0.412	0.415	6.6
ResNet (MFCC)	0.390	0.391	5.5
TDNN (MFCC)	0.250	0.254	3.0
TDNN (PLP)	0.276	0.282	3.2
TDNN_VOX (MFCC)	0.228	0.230	2.9

6.5. Results on our development test set

In Table 3, we present results on our SRE19 dev set obtained using our developed single systems in terms of NIST-SRE2019 evaluation metrics.

- TDNN_VOX - In this system, TDNN is trained on multi-style voxceleb data using MFCC features, PLDA is trained on voxceleb data and unsupervised PLDA adaptation is performed on all SRE 2018 dev plus a small portion of SRE 2018 eval data.
- ASPP (MFCC) - In this case a TDNN is augmented with an pyramidal pooling layer [12] right before statistics pooling. This system is trained on SRE's from 04 to 10, Mixer 6, Switchboard, and Fisher corpora and then fine tuned using SREs 12, 16, and 18 (all dev and a small portion of eval data). Features considered here is MFCCs.
- ResNet (MFCC) - In this case ResNet-101 is trained on SRE's from 04 to 10, Mixer 6, Switchboard, and Fisher corpora and then fine tuned using SREs 12, 16, and 18 (all dev and a small portion of eval data). Features considered here is MFCCs.
- TDNN (MFCC) & TDNN (PLP) - These systems use MFCC and PLP features, respectively. Both systems were trained on SRE's from 04 to 10, Mixer 6, Switchboard corpora. Extracted embeddings of out-of-domain data were adapted to the in-domain-data using correlation alignment-based unsupervised adaption technique [13].

7. REFERENCES

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceed-*

ings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[2] Jesús Villalba, Nanxin Chen, David Snyder, Daniel Garcia-Romero, Alan McCree, Gregory Sell, Jonas Borgstrom, Fred Richardson, Suwon Shon, François Grondin, Réda Dehak, Leibny Paola García-Perera, Daniel Povey, Pedro A. Torres-Carrasquillo, Sanjeev Khudanpur, and Najim Dehak, "State-of-the-Art Speaker Recognition for Telephone and Video Speech: The JHU-MIT Submission for NIST SRE18," in *Proc. Interspeech 2019*, 2019, pp. 1488–1492.

[3] 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.

[4] Hossein Zeinali, Shuai Wang, Anna Silnova, Pavel Matějka, and Oldřich Plchot, "BUT System Description to VoxCeleb Speaker Recognition Challenge 2019," in *VoxCeleb 2019 Workshop*, 2019.

[5] Ondřej Novotnỳ, Oldřich Plchot, Ondřej Glembek, Lukáš Burget, et al., "Analysis of DNN Speech Signal Enhancement for Robust Speaker Recognition," *Computer Speech & Language*, 2019.

[6] Ondřej Novotný, Pavel Matějka, Oldřich Plchot, and Ondřej Glembek, "On the use of DNN Autoencoder for Robust Speaker Recognition," Tech. Rep., 2018.

[7] Johan Rohdin, Themos Stafylakis, Anna Silnova, Hossein Zeinali, Lukáš Burget, and Oldřich Plchot, "Speaker verification using end-to-end adversarial language adaptation," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6006–6010.

[8] Anna Silnova, Niko Brummer, Daniel Garcia-Romero, David Snyder, and Lukáš Burget, "Fast variational bayes for heavy-tailed plda applied to i-vectors and x-vectors," in *Interspeech 2018, 19th Annual Conference of the International Speech Communication Association, Hyderabad, India, 2-6 September 2018.*, 2018.

[9] Pierre-Michel Bousquet and Mickael Rouvier, "On Robustness of Unsupervised Domain Adaptation for Speaker Recognition," in *Proc. Interspeech 2019*, 2019, pp. 2958–2962.

[10] 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.

[11] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer, "Automatic differentiation in pytorch," 2017.

[12] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 801–818.

[13] Jahangir Alam, Gautam Bhattacharya, and Patrick Kenny, "Speaker verification in mismatched conditions with frustratingly easy domain adaptation," in *Proc. Odyssey 2018 The Speaker and Language Recognition Workshop*, 2018, pp. 176–180.

[14] João Monteiro, Jahangir Alam, and Tiago H Falk, "Combining speaker recognition and metric learning for speaker-dependent representation learning," *INTERSPEECH*, 2019.

[15] David Snyder, Guoguo Chen, and Daniel Povey, "MUSAN: A Music, Speech, and Noise Corpus," 2015, arXiv:1510.08484v1.

[16] "Open Speech and Language Resources," 2017, http://www.openslr.org/28/.

[17] Daniel Garcia-Romero, Alan McCree, Stephen Shum, Niko Brummer, and Carlos Vaquero, "Unsupervised domain adaptation for i-vector speaker recognition," in *Proceedings of Odyssey: The Speaker and Language Recognition Workshop*, 2014.

8. RECIPES AT BUT

8.1. ResNet50

Shuai: The model is trained on Shanghai cluster, will migrate to but cluster in a few days (My fit account will be reserved until the end of 2020); Will be updated after preparing a clean directory

The main directory could be found in `/mnt/matylda3/xwangs01/projects/SRE19`

- Train
  - TO BE PREPARED
- Embeddings extraction
  - `eval_resnet.sh`: extract embeddings from the ResNet models.
  - `gen_h5.sh`: Convert the kaldi format xvectors to what Anya needs

8.2. F-TDNN

8.3. Res-E-TDNN + GAN

8.4. Res-E-TDNN with Denoising - Ondra Novotny

8.5. Backend - Anna Silnova

4 backend recipes are here: `/mnt/matylda5/isilnova/NIST_SRE_2019/CMN2/replicate_sre_18/recipes`

All of the needed .m files are in the same directory. The path to x-vectors is specified in the first line of the corresponding script, the directory where to save the models, scores, logs, etc. is defined in the 3rd line.

ResNet_GPLDA	`run_plda_resnet.sh`
FTDNN_HTPLDA	`run_htplda_ftdnn.sh`
Res-E-TDNN_DENOISE_HTPLDA	`run_htplda_tdnn_denoise.sh`
Res-E-TDNN_GAN_HTPLDA	`run_htplda_tdnn_gan.sh`

8.6. Calibration/Fusion - Oldrich Plchot

9. IDEAS FOR ANALYSIS

- Pavel:try again MFCC and FBANK64
- who:what

10. RETROSPECTIVE

10.1. What was good and we want to do it next time too

- who:what

10.2. What we can do better next time

- PavelM: Data processing and list preparation takes the most of the time - start way ahead
  - Shuai: We don't want this aggressive data augmentation on the disk
  - Shuai: Online data augmentation should be investigated
- who:what
- Shuai: Different pooling functions should be tried
- Shuai: More efforts targeting at the domain mismatch
- Themos: ASR-supervision.