DIHARD - CPqD - System B1/B2

Valter A. Miasato Filho[1], *Diego A. Silva*[1], *Luis Gustavo D. Cuozzo*[1]

[1]CPqD, Campinas, Sao Paulo, Brazil

{valterf,diegoa,lcuozzo}@cpqd.com.br

1. Abstract

The B1/B2 CPqD diarization system is a combination of two different neural networks, both with joint optimization for speaker embedding learning, speech activity and overlap detection. The diarization system is a simple two-stage pipeline with the network feed forward and segment generation via *k-means* speaker clustering and a combination of SAD/overlap/speaker labels for speech endpoint detection [1]. In contrast to prior work, we switched recurrent for time-convolving layers, separated SAD and speech overlap outputs, and devised a more stable training scheme by adding unit norm constraint and margin in the loss function, and batch normalization in between layers. The two networks differ by the usage of artificial prompts generated by isolated speaker datasets, as shown in section 3. Each network was chosen by the task in which they performed best in the test sets.

2. Development datasets

2.1. DIHARD development dataset

The DIHARD dataset has approximately 19 hours worth of 5-10 minute 16kHz, monaural prompts in 165 FLAC files, comprising a variety of domains shown in Table 1. The collection of all domains were split into training, validation, and test sets with the respective approximate ratios of 50%, 25% and 25%, ensuring that all domains were present under the three partitions.

2.2. Additional development datasets

We used the publicly available AMI [2] and ICSI [3] datasets as additions to the provided development data from the DIHARD challenge. Both datasets are composed of multi-party meeting recordings, from which we used the headset mixes as monaural data. The AMI corpus had poor quality in their original mix due to the noise in some channels being louder than speech in others, so we used the SoX tool [4] to apply dynamic range compression and amplitude normalization in the individual channels before mixing. We split those datasets in training, validation and test sets in roughly estimated proportions of 80%, 10% and 10%, respectively. All three partitions have disjoint sets of speakers. For augmenting our training set, we also used the Voxceleb [5] set, which annotates celebrity speech in web videos. The augmentation scheme is described in section 3.5.

2.3. DIHARD evaluation data

The evaluation data consists of around 21 hours of data with the same characteristics of the development set, except by the addition of a new domain, consisting of recordings from conversations in restaurants. The same set was used in two different tracks for the challenge: diarization from gold speech segmentation (Track1) and diarization from scratch (Track2).

Table 1: *Development datasets.*

Domain	Duration	Speech%	Ovp%	Spk#
AMI	75:39:25	85.8	16.3	3 to 6
ICSI	71:41:12	85.6	15.2	3 to 10
DIHARD	18:56:50	76.1	6.3	1 to 10
SEEDLINGS	1:50:58	60.1	9.3	2 to 5
SCOTUS	2:04:46	84.0	1.6	5 to 10
DCIEM	2:29:58	68.5	2.0	2
ADOS	2:10:12	61.0	2.3	2 to 3
YP	2:03:25	78.5	1.0	3 to 5
SLX	2:00:26	72.4	5.7	2 to 6
VAST	1:50:20	85.7	11.8	1 to 9
RT04S	2:26:15	93.7	21.7	3 to 10
LIBRIVOX	2:00:30	79.4	0.0	1

3. Algorithm description

This section describes the algorithm for both networks, except for section 3.4, which is only applicable for the B2 model, and for section 3.6, where the usage of both is described.

3.1. Affinity matrix loss

The custom loss for speaker embedding generation, which is inspired by [6] and [1], is defined by the following equation:

$$C(Y,V) = \sum_{i=1}^{T}\sum_{j=1}^{T} \max((|y_i|v_i \bullet |y_j|v_j - y_i \bullet y_j)^2 - m, 0). \tag{1}$$

In which T is the number of timesteps from each training example, y_i is the one-hot representation of speakers in the frame i and v_i is the embedding to be learned for the same frame. The affinity matrix loss is used for learning the embedding output depicted in Figure 1.

3.2. Neural network topology

Our network is comprised of seven time-convolving layers. Each layer is described in Figure 1, with w standing for the width of the convolution and d for the dilation. All layers have $D = 512$ filters for convolution, and have *ReLUs* as nonlinearities. Batch normalization [7] is applied in between layers for more stable training and faster convergence. To avoid fine-tuning gradient descent parameters, the Adam optimizer [8] was employed and gradients were clipped for having the maximum norm of 1.

The outputs of the network are all connected to the last layer with time-distributed weights. The embedding output is a fully connected layer with $K = 100$ activations constrained

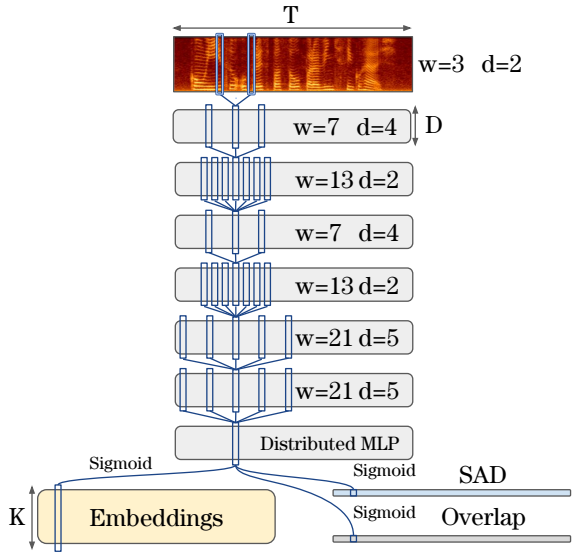

Figure 1: *Neural network topology.*

by the sigmoid non-linearity. The final vectors are then divided by their norm. The SAD and overlap outputs are both single sigmoids for binary classification.

3.3. Training

The input of our network is the log spectrum of the audio prompts in which speaker diarization is to be performed. We chose a window of $25ms$ with a shift of $30ms$ to perform the short-time Fourier transform. This configuration was inspired by [9] and was used for faster learning and inference. The block size in number of timesteps was $T = 1024$, which accounts for roughly $30s$ of context.

For balancing speech activity and overlap data, we apply sample weights based on a running ratio of the amount of positive/negative examples. The margin value for the affinity matrix loss was set to $m = 0.2$.

We sample 512 batches of 64 examples from different files through 200 iterations, each taking an average of $3880s$ to complete. Intermediate models are saved with SAD, overlap and DER metrics and we choose the best in each task to compose the final system.

3.4. Artificial prompt generation

For generating the artificial prompts with the VoxCeleb [5] dataset, we used a simple scheme of sampling at most 2 utterances from a range of 3 to 10 speakers from the dataset. The utterances were concatenated in prompts of length T and the SAD/overlap outputs weights were set to 0 in these examples. Only the B2 system used these examples for training.

3.5. Data augmentation

We applied two data augmentation techniques for our datasets: noise addition in the AMI and ICSI datasets and noise suppression in the samples provided for the DIHARD challenge. The noise addition was performed with the FaNT tool [10], using external noise samples. We applied CHIME3 [11] samples over AMI, and QUT-NOISE-TIMIT [12] samples over ICSI, both with random signal-to-noise ratios between 5dB and 15dB. The noise supression was used with the corresponding module from the WebRTC project [13] in the DIHARD development set.

3.6. Diarization system

For the track 2 submission, we used the best iteration in terms of SAD accuracy from the B1 system for generating the SAD labels (only in track 2), and the best overall DER from the B2 system to generate the overlap labels and embeddings. We run the mini-batch *k-means* for 2 to 10 clusters and choose the optimal speaker count by using the silhouette score. All outputs are then used in the speech endpoint algorithm from [1] to generate the final segments. The window size for the algorithm was set to $30ms$, which is the duration of a single frame from our STFT.

4. Hardware description and timing

The models were trained on a 32 Intel® Xeon® E5-2686 @2.30GHz machine with Ubuntu OS 16.04 equipped with eight instances of the NVIDIA Tesla K80 Graphics Processing Units over Amazon (AWS) p2.8xlarge instance, with 480GB of available RAM. The training and development process was based on the Keras framework with Tensorflow backend and NVIDIA CUDA® 9.0 version. The floating point precision for running the experiments was the default 32-bit precision from the toolkits.

In training time, each model was run on a single GPU, with the feature extraction and batch generation steps processed on shared CPUs. The GPU time was observed as the bottleneck of the process. The total training time for each model was roughly 9 days, with the possibility of training a total of 8 systems at the same time.

The benchmarked inference time over the full evaluation set was computed over GPU processing. The usage of GPU in this case was arguably suboptimal. To leverage its computing power in our pipeline, we chose to generate features for the full duration of a single file prior to forwarding it through the network. For processing the full pipeline over the evaluation dataset, it took 26 minutes, from which 24 minutes were needed for network feed-forwards (12 minutes each), and 2 minutes were needed for 9 mini-batch *k-means* runs, silhouette score computation and speech endpoint detection algorithm. Each stage was run in all files from the evaluation set before running the next one.

For benchmarking a single file, we chose a regular desktop machine with a dual-core Intel® Core™ i3-6100 @3.7 GHz with 8GB of RAM. In this scheme, we generated features and feed-forwarded them on demand through the networks to avoid excessive RAM consumption. Our test file had 44 minutes of duration. The intermediate feature files held 32MiB worth of disk storage. The full pipeline took 3min 00s with a peak RAM consumption of 1250MiB.

5. References

[1] V. A. Miasato Filho, D. A. Silva, and L. G. D. Cuozzo, "Multi-objective long-short term memory neural networks for speaker diarization in telephone interactions," in *2017 Brazilian Conference on Intelligent Systems (BRACIS)*. IEEE, 2017, pp. 181–185.

[2] J. Carletta, S. Ashby, S. Bourban, M. Flynn, M. Guillemot, T. Hain, J. Kadlec, V. Karaiskos, W. Kraaij, M. Kronenthal *et al.*, "The ami meeting corpus: A pre-announcement," in *International Workshop on Machine Learning for Multimodal Interaction*. Springer, 2005, pp. 28–39.

[3] A. Janin, D. Baron, J. Edwards, D. Ellis, D. Gelbart, N. Morgan, B. Peskin, T. Pfau, E. Shriberg, A. Stolcke *et al.*, "The icsi meeting corpus," in *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, vol. 1. IEEE, 2003, pp. I–I.

[4] L. Norskog and C. Bagwell, "Sox-sound exchange," http://sox.sourceforge.net/, 2013, version 14.

[5] A. Nagrani, J. S. Chung, and A. Zisserman, "Voxceleb: a large-scale speaker identification dataset," *arXiv preprint arXiv:1706.08612*, 2017.

[6] J. R. Hershey, Z. Chen, J. Le Roux, and S. Watanabe, "Deep clustering: Discriminative embeddings for segmentation and separation," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 31–35.

[7] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, 2015, pp. 448–456.

[8] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[9] D. Povey, V. Peddinti, D. Galvez, P. Ghahrmani, V. Manohar, X. Na, Y. Wang, and S. Khudanpur, "Purely sequence-trained neural networks for ASR based on lattice-free MMI," *Submitted to Interspeech*, 2016.

[10] H. G. Hirsch, "Fant: filtering and noise adding tool," *Niederrhein University of Applied Sciences, http://dnt.-kr. hsnr. de/download. html*, 2005.

[11] J. Barker, R. Marxer, E. Vincent, and S. Watanabe, "The third 'chime'speech separation and recognition challenge: Dataset, task and baselines," in *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*. IEEE, 2015, pp. 504–511.

[12] D. B. Dean, S. Sridharan, R. J. Vogt, and M. W. Mason, "The qut-noise-timit corpus for the evaluation of voice activity detection algorithms," *Proceedings of Interspeech 2010*, 2010.

[13] A. B. Johnston and D. C. Burnett, *WebRTC: APIs and RTCWEB protocols of the HTML5 real-time web*. Digital Codex LLC, 2012.