Universidad de los Andes at ActivityNet Challenge 2020 - Task B Active Speaker Detection (AVA)

Juan León Alcázar[1], Pablo Arbeláez[1]
[1]Universidad de los Andes,
[1]{jc.leon,pa.arbelaez}@uniandes.edu.co;

1. Active Speakers in Context

At its core, our strategy estimates an active speaker score for an individual face (target face) by analyzing the target itself, the current audio input, and multiple faces detected at the current timestamp. We begin by analyzing audiovisual information from short video clips. The visual information is a stack of k consecutive face crops sampled from a time interval τ. The audio information is the raw waveform sampled over the same τ interval. For every clip $c_{s,\tau}$ in a video sequence, we compute an embedding $\mathbf{u}_{s,\tau}$ using a short-term encoder $\Phi(c_{s,\tau})$.

Short-term Encoder (Φ). We approximate Φ by means of a two-stream convolutional architecture. The visual stream takes as input a tensor $\mathbf{v} \in \mathbb{R}^{H\times W\times(3k)}$, where H and W are the width and height of k face crops. On the audio stream, we convert the raw audio waveform into a Mel-spectrogram represented as $\mathbf{a} \in \mathbb{R}^{Q\times P}$, where Q and P depend on the length of the interval τ. On a forward pass the visual sub-network estimates a visual embedding $\mathbf{u_v} \in \mathbb{R}^{d_v}$, while the audio sub-network computes an audio embedding $\mathbf{u_a} \in \mathbb{R}^{d_a}$. We compose an audiovisual feature embedding $\mathbf{u} \in \mathbb{R}^{d}$ by concatenating the output embedding of each stream.

Structured Context Ensemble. Once the clip features $\mathbf{u} \in \mathbb{R}^{d}$ have been estimated, we proceed to assemble these features into a set that encodes contextual information. We denote this set as the Active Speaker Ensemble. To construct this ensemble, we first define a long interval T ($T > \tau$) centered at a reference time t, and designate one of the speakers present at t as the reference speaker and every other speaker is designated as context speaker. This sampling scheme yields a tensor $\mathbf{C}_t$ with dimensions $L \times S \times d$, where S is the total number of speakers analyzed. Figure 1 contains a detailed example on the sampling process.

1.1. Context Refinement

Our model the Active Speaker Context (ASC) consists of two core components. First, it implements a multi-modal self-attention mechanism to establish pairwise interactions between the audiovisual observations on $\mathbf{C}_t$. Second, it incorporates a long-term temporal encoder, which exploits temporal structures in conversations.

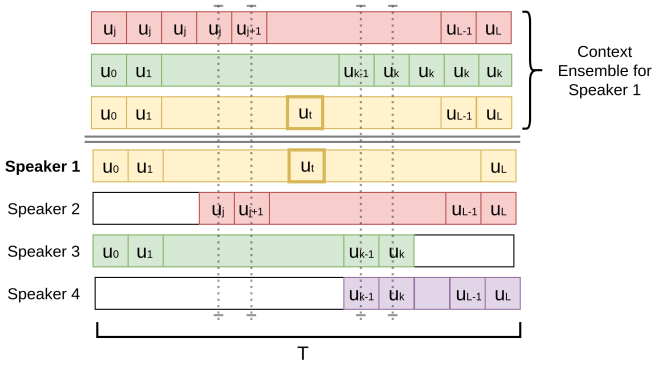

Figure 1. **Building Context Tensors.** We build a context ensemble given a reference speaker (Speaker 1 in this example), and a reference time t. First, we define a long-term sampling window T containing $L+1$ clips centered at time t, $T = \{0, 1, ..., t, ..., L-1, L\}$. We select as context speakers those that overlap with the reference speaker at t (speakers 2 and 3). Finally, we sample clip-level features u_l throughout the whole sampling window T from the reference speaker and all the speakers designated as context. If the temporal span of the speaker does not entirely match the interval T, we pad it with the initial or final speaker features. For instance, Speaker 2 is absent between 0 and i, so we pad left with u_i. Similarly, for speaker 3, we pad right with u_k. Notice that, by our definition, Speakers 2 and 3 could switch positions, but Speaker 1 must remain at the bottom of the stack.

Pairwise Refinement. We start from the multi-modal context ensemble $\mathbf{C}_t$, and model pairwise affinities between observations in $\mathbf{C}_t$ regardless of their temporal order or the speaker they belong to. We do this refinement by following a strategy similar to Vaswani *et al.* [8]. In practice, we adapt the core idea of pair-wise attention from the non-local framework [9] to work over multi-modal high-level features, thereby estimating a dense attention map over the full set of clips contained in the sampling window T.

Temporal Refinement. The goal of this long-term pooling step is two-fold. First, to refine the weighted features in $\mathbf{C}_t^{\dagger}$ by directly attending to their temporal structure. Second, to reduce the dimensionality of the final embedding to d' ($d > d'$), allowing us to use a smaller fully-connected prediction layer. Given the inherent sequential structure of the task, we implement this refinement using an LSTM model [4].

2. Training and Implementation

We use a two-stream (visual and audio) convolutional encoder based on the Resnet-18 architecture [3] for the Short-Term Feature extraction (STE). Following [7], we repurpose the video stream to accept a stack of 11 face crops by replicating the weights on the input layer 11 times. The audio stream input is a Mel-spectrogram calculated from an audio snippet, which exactly matches the time interval covered by the visual stack. Since Mel-spectrograms are 2D tensors, we re-purpose the input of the audio stream to accept a $L \times P \times 1$ tensor by averaging channel-specific weights at the input layer.

Training the Short-term Encoder We train the STE using the Pytorch library [6] for 100 epochs. We choose the ADAM optimizer with an initial learning rate of 3×10^{-4} and learning rate annealing $\gamma = 0.1$ every 40 epochs. We resize every face crop to 124×124 and perform random flipping and corner cropping uniformly along the visual input stack. We drop the large-scale multi-modal pre-training of [1], in favor of standard Imagenet [2] pre-training for the initialization.

Since we want to favor the estimation of discriminative features on both streams, we follow the strategy presented by Roth *et al*. [7] and add two auxiliary supervision sources, and place them on top of both streams before the feature fusion operation, this creates two auxiliary loss functions $\mathcal{L}_a, \mathcal{L}_v$. Our final loss function is $\mathcal{L} = \mathcal{L}_{av} + \mathcal{L}_a + \mathcal{L}_v$. We use the standard Cross-entropy loss for all three terms.

Training the Active Speaker Context Model We also optimize the ASC using the Pytorch library and the ADAM optimizer with an initial learning rate of 3×10^{-6} and learning rate annealing $\gamma = 0.1$ every 10 epochs. We train the full ASC module from scratch and include batch normalization layers to favor faster convergence [5]. Similar to the STE, we use Cross-entropy loss to train ASC, but in this scenario, the loss consists of a single term $\mathcal{L}_{av}$. The ASC processes a fixed number of speakers $S = 3$.

As Table 1 shows, our method achieves competitive results in the testing subset. Even though our model discards 3D convolutions and model ensembles [1], we rank 2nd in the AVA-ActiveSpeaker 2019 Leaderboard The overall results on the AVA-ActiveSpeaker validation and testing subsets validate the effectiveness of our approach.

Method	**mAP**
Validation subset	
Active Speakers Contex (Ours)	**87.1**
Chung *et al*. (Temporal Convolutions) [1]	85.5
Chung *et al*. (LSTM) [1]	85.1
Zhang *et al*. [10]	84.0
ActivityNet Challenge Leaderboard 2019	
Naver Corporation [1]	87.8
Active Speakers Context (Ours)	**86.7**
University of Chinese Academy of Sciences [10]	83.5
Google Baseline [7]	82.1

Table 1. **Comparison with the State-of-the-art.** We report the performance of state-of-the-art methods in the AVA Active Speakers validation and testing subsets. Results in the validation set are obtained using the official evaluation tool published by [7], test set metrics are obtained using the the ActivityNet challenge evaluation server. In the validation subset, we improve the performance of previous approaches by 1.6%, without using large-scale multimodal pre-training. In the test subset, we achieve 86.7% and rank second in the leaderboard, without using 3D convolutions, sophisticated post-processing heuristics or assembling multiple models.

References

[1] Joon Son Chung. Naver at activitynet challenge 2019–task b active speaker detection (ava). *arXiv preprint arXiv:1906.10555*, 2019.

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[4] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[5] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[6] 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. In *NeurIPS-Workshop*, 2017.

[7] Joseph Roth, Sourish Chaudhuri, Ondrej Klejch, Radhika Marvin, Andrew Gallagher, Liat Kaver, Sharadh Ramaswamy, Arkadiusz Stopczynski, Cordelia Schmid, Zhonghua Xi, et al. Ava-activespeaker: An audio-visual dataset for active speaker detection. *arXiv preprint arXiv:1901.01342*, 2019.

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[9] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018.

[10] Yuan-Hang Zhang, Jingyun Xiao, Shuang Yang, and Shiguang Shan. Multi-task learning for audio-visual active speaker detection.