CROSS ATTENTIVE POOLING FOR SPEAKER VERIFICATION

Seong Min Kye1,2, Yoohwan Kwon1,3, Joon Son Chung1

1Naver Corporation, 2Korea Advanced Institute of Science and Technology, 3Yonsei University

ABSTRACT

The goal of this paper is text-independent speaker verification where utterances come from ‘in the wild’ videos and may contain irrelevant signal. While speaker verification is naturally a pair-wise problem, existing methods to produce the speaker embeddings are instance-wise. In this paper, we propose Cross Attentive Pooling (CAP) that utilizes the context information across the reference-query pair to generate utterance-level embeddings that contain the most discriminative information for the pair-wise matching problem. Experiments are performed on the VoxCeleb dataset in which our method outperforms comparable pooling strategies.

Index Terms: speaker recognition, speaker verification, cross attention.

1. INTRODUCTION

Automatic speaker recognition is an attractive way to verify someone’s identity since the voice of a person is one of the most easily accessible biometric information. Due to this non-invasive nature and the technological progress, speaker recognition has recently gained considerable attention both in the industry and in research.

While the definition of speaker recognition encompasses both identification and verification, the latter has more practical applications – for example, the use of speaker verification is becoming popular in call centres and in AI speakers. Unlike closed-set identification, open-set verification aims to verify the identity of speakers unseen during training. Therefore, speaker verification is naturally a metric learning problem in which voices must be mapped to representations in a discriminative embedding space.

While mainstream literature in the field have learnt speaker embeddings via the classification loss [1, 2, 3, 4], such objective functions are not designed to optimize embedding similarity. More recent works [5, 6, 7, 8, 9, 10, 11] have used additive margin variants of the softmax function [12, 13, 14] to enforce inter-class separation which has been shown to improve verification performance.

Since open-set verification addresses identities unseen during training, it can be formulated as a few-shot learning problem where the network should recognize unseen classes with limited examples. Prototypical networks [15] have been proposed in which the training mimics the few-shot learning scenario, and this strategy has recently shown to achieve competitive performance in speaker verification [16, 17, 18, 19, 20].

In order to train networks to optimise the similarity metric, frame-level representations produced must first be aggregated into an utterance-level embedding. A naïve way to produce an utterance-level embedding is to take a uniformly weighted average of the frame-level representations, which is referred to as Temporal Average Pooling (TAP) in the existing literature. Self-Attentive Pooling (SAP) [21] has been proposed to pay more attention to the frames that are more discriminative for verification. However, the instance-level self-attention finds the features that are more discriminative for speaker verification in general (i.e. across the whole training set) rather than for the specific examples in the support set.

In few-shot learning, cross attention networks (CAN) [22] has been recently proposed to select attention based on unseen target classes, by attending to the parts of the input image that is relevant and discriminative to the examples in the support set. This idea is applicable to speaker verification, since the features that are discriminative for comparing an utterance against one class (speaker) in the support set may be different to the features for comparing to another class.

To this end, we propose cross attentive pooling (CAP) which computes the attention with reference to the example in the support set in order to effectively aggregate frame-level information into an utterance level embedding. In this way, the network is able to identify and focus on the parts of the utterance that provide characterising features for the particular class in the support set. This is similar to how humans tend to look for common characterising features between the pair of samples when recognising instances from unseen classes, whether these are speakers or visual objects. Unlike instance-level pooling, the proposed attention module takes full advantage of the pair-wise nature of the verification task, by modelling the relevance between the class (prototype) feature and the query feature.

The effectiveness of our method is demonstrated on the popular VoxCeleb dataset [23] in which we report improvements over existing pooling methods.
2. METHODS

2.1. Few-shot learning framework

We use a few-shot learning framework in order to train the embeddings for speaker recognition. In particular, our implementation is based on prototypical networks [15], which has been shown to perform well in speaker verification [17] [18] [19].

**Batch formation.** Each mini-batch contains a support set $\mathcal{S}$ and a query set $\mathcal{Q}$. A mini-batch contains $M$ utterances from each of $N$ different speakers. We use a single utterance for each speaker in the support set $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^{N \times 1}$ and the rest of the utterances ($2 \leq i \leq M$) in the query set $\mathcal{Q} = \{(\tilde{x}_i, \tilde{y}_i)\}_{i=1}^{N \times (M-1)}$, where $y, \tilde{y} \in \{1, \ldots, N\}$ is the class label.

**Training objective.** Since the support set is formed with a single utterance $x$, the prototype (or centroid) is the same as the support utterance for each speaker $y$:

$$P_y = x$$

The cross-entropy loss with a log-softmax function is used to minimise the distance between segments from same speaker and maximise the distance between different speakers.

$$L_{NP} = -\frac{1}{|\mathcal{Q}|} \sum_{(\tilde{x}, \tilde{y}) \in \mathcal{Q}} \log \sum_{y=1}^{N} e^{d(\tilde{x}, P_y)} \tag{2}$$

We use the same distance metric as [16], where the distance function is the cosine similarity between the prototype and the query with the scale of the query embedding.

$$d(\tilde{x}, P_y) = \frac{\tilde{x}^T P_y}{\|P_y\|_2} = \|\tilde{x}\|_2 \cdot \cos(\tilde{x}, P_y) \tag{3}$$

We refer to the prototypical loss with this similarity function as the Normalised Prototypical (NP) loss in the rest of this paper.

Kye et al. [16] has used episodic training together with a global classification loss in order to make speaker embeddings more discriminative. Global classification is applied to both the support and the query sets. By incorporating the softmax classification loss, we can train the embeddings to be discriminative over all classes, as opposed to only classes in the mini-batch. The final objective is the sum of NP and the softmax cross-entropy losses with equal weighting.

$$P_y = x$$

2.2. Instance-wise aggregation

An ideal utterance-level embedding should be invariant to temporal position, but not frequency. Since 2D convolutional neural networks [24] [25] produce 2D activation maps, [1] has proposed aggregation layers that are fully connected only along the frequency axis. This produces a $1 \times T$ feature map before the pooling layers, which are described in the following sections.

**Temporal Average Pooling (TAP).** The TAP layer simply takes the mean of the features along the time domain.

$$e = \frac{1}{T} \sum_{n=1}^{T} x_t \tag{4}$$

**Self-Attentive Pooling (SAP).** In contrast to the TAP layer that pools the features over time with uniform weights, the self-attentive pooling (SAP) layer [21] [26] [27] pays attention to the frames that are more informative for utterance-level speaker recognition.

$$h_t = \tanh(W x_t + b) \tag{5}$$

The similarity between the hidden vectors and a learnable context vector $\mu$ is computed, which represents the relative importance of the hidden feature. The context vector can be seen as a high-level representation of what makes the frames informative for speaker recognition.

$$w_t = \frac{\exp(h_t^T \mu)}{\sum_{t=1}^{T} \exp(h_t^T \mu)} \tag{6}$$

The utterance-level embedding $e$ can be obtained as a weighted sum of the frame-level representations.

$$e = \sum_{t=1}^{T} w_t x_t \tag{7}$$

2.3. Pair-wise aggregation

Unlike traditional instance-wise aggregation, our proposed method aggregates frame-level features, utilizing information of the frame features of the other utterance. In order to match the objective in training and testing, we use the prototypical networks [15], which is metric-based meta-learning framework. In this framework, we train our cross attentive pooling (CAP) using the pairs of support and query set. In the test scenario, support set and query set correspond to enrollment and test utterances, respectively.

For every pair of utterances from the query and the support sets, we extract frame-level representations $s = \{s_1, s_2, \ldots, s_T\}$ and $q = \{q_1, q_2, \ldots, q_T\}$. Then, with the meta-projection layer $g_{\phi}(\cdot)$, we extract hidden features from the frame-level representation. This non-linear projection allows us to quickly adapt to an arbitrary frames, so that the similarity of the frame pair can be well measured. The
layer consists a single-layer perceptron followed by a ReLU activation function.

\[ g_\phi(\cdot) = \max(0, W(\cdot) + b) \quad (8) \]

After the meta-projection layer, we can obtain \( S = \{S_i\}_{i=1}^{T_s}\) and \( Q = \{Q_i\}_{i=1}^{T_q}\) as hidden representations for every frame, where \( S_i \) and \( Q_i \) denotes \( g_\phi(s_i) \) and \( g_\phi(q_i) \), respectively.

**Correlation matrix.** Correlation matrix \( R \) summarises similarity for every possible pair of frames. \( R^S \in \mathbb{R}^{T_s \times T_q} \) is computed as:

\[ R^S_{i,j} = \left( S_i \| S_i \|_2 \right)^T \left( Q_j \| Q_j \|_2 \right) \quad (9) \]

Note that \( R^Q = (R^S)^T \).

**Pair-adaptive attention.** In order to obtain the pair-adaptive context vector, we average correlation matrix along with its own time axis as follows:

\[ \mu_s = \frac{1}{T_s} \sum_{i=1}^{T_s} R^S_{i,i} \quad (10) \]

where \( \mu_s \in \mathbb{R}^{T_q} \) and \( R^S_{i,i} \) denotes \( i \)-th row vector. Each row vector has the information of similarity to all frames of the other utterance. Therefore, the average correlation for each frame of the other utterance can be presented by \( \mu \), which is used in the context vector to determine how similar it is to the other utterance.

The attention weights are given by the following equation for every utterance.

\[ w^s_t = \frac{\exp(\mu^T s R^S_{t,:})/\tau)}{\sum_{i=1}^{T_s} \exp(\mu^T s R^S_{i,:})/\tau)} \quad (11) \]

where \( \tau \) is temperature scaling, which sharpens attention distribution.

\[ e_s = \frac{1}{T_s} \sum_{t=1}^{T_s} (1 + w^s_t) s_t \quad (12) \]

As done in Hou et al. [22], we use a residual attention mechanism to obtain the utterance-level feature. For the other utterance, the utterance-level feature of \( q \), \( e_q \) can be obtained in the same way.

### 3. EXPERIMENTS

#### 3.1. Input representations

During training, we randomly extract fixed length 2-second temporal segments from each utterance. Spectrograms are extracted with a hamming window of width 25ms and step 10ms. 40-dimensional Mel filterbanks are used as the input to the network. Mean and variance normalisation (MVN) is performed with instance normalisation [28]. Since the VoxCeleb dataset contains continuous speech, voice activity detection (VAD) is not used during training and testing.

#### 3.2. Trunk architecture

Experiments are performed using the Fast ResNet-34 architecture introduced in [19].

Residual networks [25] are used widely in image recognition and has recently been applied to speaker recognition [6].
Fast ResNet-34 is the same as the original ResNet with 34 layers, except with only one-quarter of the channels in each residual block in order to reduce computational cost. The model only has 1.4 million parameters compared to 22 million of the standard ResNet-34, and minimises the computation cost by reducing the activation maps early in the network. The network architecture is given in Table 1.

Table 1. Fast ResNet-34 architecture. ReLU and batchnorm layers are not shown. Each row specifies the number of convolutional filters, their sizes and strides as size × size, # filters, stride. The output from the fully connected layer is ingested by the pooling layers.

| layer name | Filters | Output |
|------------|---------|--------|
| conv1      | \(7 \times 7, 16, \text{stride } 2\) \(\times 3, \text{Maxpool, stride } 2\) | \(20 \times T \times 16\) |
| conv2      | \(3 \times 3, 16\) \(3 \times 3, 16\) \(\times 3, \text{stride } 1\) | \(20 \times T \times 16\) |
| conv3      | \(3 \times 3, 32\) \(3 \times 3, 32\) \(\times 4, \text{stride } 2\) | \(10 \times T/2 \times 32\) |
| conv4      | \(3 \times 3, 64\) \(3 \times 3, 64\) \(\times 6, \text{stride } 2\) | \(5 \times T/4 \times 64\) |
| conv5      | \(3 \times 3, 128\) \(3 \times 3, 128\) \(\times 3, \text{stride } 2\) | \(5 \times T/4 \times 128\) |
| pool       | \(9 \times 1\) | \(1 \times T/4 \times 128\) |
| aggregation| TAP or SAP or CAP | \(1 \times 128\) |
| fc         | FCN, 512 | \(1 \times 512\) |

3.3. Implementation details

Datasets. The networks are trained on the development set of VoxCeleb2 [29] and tested on the original test set of VoxCeleb1 [11]. Note that there is no overlap between the development set of VoxCeleb2 dataset and the VoxCeleb1 dataset.

Training. Our implementation is based on the PyTorch framework [30]. The models are trained using a NVIDIA V100 GPU with 32GB memory for 500 epochs. The networks are trained with the Adam optimizer, and we use an initial learning rate of 0.001 with a decay of 5% every 10 epochs. We use a fixed batch size of 200 for all experiments. The networks take 2 days to train using a single GPU.

Data augmentation. Aside from taking random 2-second segments, no data augmentation is performed during training or testing.

3.4. Evaluation

Evaluation protocol. We report two performance metrics: (1) the Equal Error Rate (EER) which is the rate at which both acceptance and rejection errors are equal; and (2) the minimum of the detection cost function function used by the NIST SRE [31] and the VoxCeleb Speaker Recognition Challenge (VoxSRC) [2] evaluations. In order to compute the EER, we sample 10 3.5-second speech segments at regular time intervals from each utterance and compute the mean of \(10 \times 10 = 100\) distances from all possible combinations per each pair. This protocol is in line with that used by [29, 20]. The parameters \(C_{\text{miss}} = 1\), \(C_{\text{fa}} = 1\) and \(P_{\text{target}} = 0.05\) are used for the cost function, same as that used in the VoxSRC.

Table 2. Comparison with various aggregation methods. † Note that [16] uses the same ResNet-34 network but with twice as many filters in all layers. NP: Normalised Prototypical, AP: Angular Prototypical, TAP: Temporal Average Pooling, SAP: Self-Attentive Pooling, CAP: Cross-Attentive Pooling.

| Loss Aggregation | MinDCF | EER (%) |
|------------------|--------|---------|
| NP + Softmax [16]† | TAP | - | 2.22 |
| NP + Softmax | TAP | - | 2.08 |
| NP + Softmax | CAP | 0.164 | 2.13 |
| NP + Softmax | CAP | 0.161 | 2.08 |
| NP + Softmax | CAP | 0.143 | 1.93 |

Results. The results are given in Table 2. The baseline results are in line with those reported by previous work using comparable methods and architecture. Cross-Attentive Pooling outperforms existing methods on the popular VoxCeleb dataset, and by a significant margin using the MinDCF measure. It should be noted that the result outperforms all existing work on the dataset that use a model size similar to ours (1.4 million parameters).

4. CONCLUSION

In this paper, we presented pair-wise cross attentive pooling method for speaker verification. In contrast to the instance-based methods, the pair-wise strategy benefits from the contextual information by looking at the parts of speech pair. The pair-wise pooling method is not only applicable to the prototypical framework, but also to other metric learning objectives such as the contrastive loss.

Acknowledgements. We would like to thank Bong-Jin Lee and Icksang Han for helpful discussions.
5. REFERENCES

[1] Arsha Nagrani, Joon Son Chung, and Andrew Zisserman, “VoxCeleb: a large-scale speaker identification dataset,” in NeurIPS, 2018.

[2] David Snyder, Daniel Garcia-Romero, Daniel Povey, and Sanjeev Khudanpur, “Deep neural network embeddings for text-independent speaker verification,” in Interspeech, 2017, pp. 999–1003.

[3] David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur, “X-vectors: Robust dnn embeddings for speaker recognition,” in Proc. ICASSP. IEEE, 2018, pp. 5329–5333.

[4] Yoo-hwan Kwon, Soo-Whan Chung, and Hong-Goo Kang, “Intra-class variation reduction of speaker representation in disentanglement framework,” arXiv preprint arXiv:2008.01348, 2020.

[5] Youngmoon Jung, Seong Min Kye, Yeunju Choi, Myunghun Jung, and Hoirin Kim, “Improving multi-scale aggregation using feature pyramid module for robust speaker verification of variable-duration utterances,” in Interspeech, 2020.

[6] Weidi Xie, Arsha Nagrani, Joon Son Chung, and Andrew Zisserman, “Utterance-level aggregation for speaker recognition in the wild,” in Proc. ICASSP, 2019.

[7] Mahdi Hajibabaei and Dengxin Dai, “Unified hypersphere embedding for speaker recognition,” arXiv preprint arXiv:1807.08312, 2018.

[8] Yi Liu, Liang He, and Jia Liu, “Large margin softmax loss for speaker verification,” in Interspeech, 2019.

[9] Daniel Garcia-Romero, David Snyder, Gregory Sell, Alan McCree, Daniel Povey, and Sanjeev Khudanpur, “X-vector dnn refinement with full-length recordings for speaker recognition,” in Interspeech, 2019, pp. 1493–1496.

[10] Hossein Zeinali, Shuai Wang, Anna Silnova, Pavel Matějka, and Oldřich Plebhot, “BUT system description to VoxCeleb Speaker Recognition Challenge 2019;” arXiv preprint arXiv:1910.12592, 2019.

[11] Xu Xiang, Shuai Wang, Houjun Huang, Yanmin Qian, and Kai Yu, “Margin matters: Towards more discriminative deep neural network embeddings for speaker recognition,” in Asia-Pacific Signal and Information Processing Association Annual Summit and Conference, 2019.

[12] Feng Wang, Jian Cheng, Weiyang Liu, and Haidun Liu, “Adaptive margin softmax for face verification,” IEEE Signal Processing Letters, vol. 25, no. 7, pp. 926–930, 2018.

[13] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu, “Cosface: Large margin cosine loss for deep face recognition,” in Proc. CVPR, 2018, pp. 5265–5274.

[14] Jiankang Deng, Jia Guo, Nianman Xue, and Stefano Zafeiriou, “Arcface: Additive angular margin loss for deep face recognition,” in Proc. CVPR, 2019, pp. 4690–4699.

[15] Jake Snell, Kevin Swersky, and Richard Zemel, “Prototypical networks for few-shot learning,” in NeurIPS, 2017, pp. 4077–4087.

[16] Seong Min Kye, Youngmoon Jung, Hae Beom Lee, Sung Ju Hwang, and Hoirin Kim, “Meta-learning for short utterance speaker recognition with imbalance length pairs,” in Interspeech, 2020.

[17] Jixuan Wang, Kuan-Chieh Wang, Marc T Law, Frank Rudzicz, and Michael Brudno, “Centroid-based deep metric learning for speaker recognition,” in Proc. ICASSP. IEEE, 2019, pp. 3652–3656.

[18] Prashant Anand, Ajeet Kumar Singh, Siddharth Srivastava, and Brejesh Lall, “Few shot speaker recognition using deep neural networks,” arXiv preprint arXiv:1904.08775, 2019.

[19] Joon Son Chung, Jaesung Huh, Seongkyu Mun, Minjae Lee, Hee Soo Heo, Soyeon Choe, Chiheon Ham, Sunghwan Jung, Bong-Jin Lee, and Icksang Han, “In defence of metric learning for speaker recognition,” in Interspeech, 2020.

[20] Jaesung Huh, Hee Soo Heo, Jingu Kang, Shinji Watanabe, and Joon Son Chung, “Augmentation adversarial training for unsupervised speaker recognition,” arXiv preprint arXiv:2007.12085, 2020.

[21] Weicheng Cai, Jinkun Chen, and Ming Li, “Exploring the encoding layer and loss function in end-to-end speaker and language recognition system,” in Speaker Odyssey, 2018.

[22] Ruibing Hou, Hong Chang, MA Bingpeng, Shiguang Shan, and Xilin Chen, “Cross attention network for few-shot classification,” in NeurIPS, 2019, pp. 4003–4014.

[23] Arsha Nagrani, Joon Son Chung, Weidi Xie, and Andrew Zisserman, “Voxceleb: Large-scale speaker verification in the wild,” Computer Speech and Language, 2019.

[24] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, “Return of the devil in the details: Delving deep into convolutional nets,” in Proc. BMVC, 2014.

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, “Deep residual learning for image recognition,” in Proc. CVPR, 2016.

[26] Gautam Bhattacharya, Md Jahangir Alam, and Patrick Kenny, “Deep speaker embeddings for short-duration speaker verification,” in Interspeech, 2017, pp. 1517–1521.

[27] FA Rezaur rahman Chowdhury, Quan Wang, Ignacio Lopez Moreno, and Li Wan, “Attention-based models for text-dependent speaker verification,” in Proc. ICASSP. IEEE, 2018, pp. 5359–5363.

[28] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky, “Instance normalization: The missing ingredient for fast stylization,” arXiv preprint arXiv:1607.08022, 2016.

[29] Joon Son Chung, Arsha Nagrani, and Andrew Zisserman, “VoxCeleb2: Deep speaker recognition,” in INTERSPEECH, 2018.

[30] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al., “Pytorch: An imperative style, high-performance deep learning library,” in NeurIPS, 2019, pp. 8024–8035.

[31] NIST 2018 Speaker Recognition Evaluation Plan, 2018 (accessed 31 July 2020), https://www.nist.gov/system/files/documents/2018/08/17/sre18_eval_plan_2018-05-31_v6.pdf See Section 3.1.