GRAPH ATTENTIVE FEATURE AGGREGATION
FOR TEXT-INDEPENDENT SPEAKER VERIFICATION

Hye-jin Shim[1†], Jungwoo Heo[1†], Jae-han Park[2], Ga-hui Lee[2], and Ha-Jin Yu[1]*

[1]School of Computer Science, University of Seoul, [2]KT Corporation

ABSTRACT

The objective of this paper is to combine multiple frame-level features into a single utterance-level representation considering pairwise relationships. For this purpose, we propose a novel graph attentive feature aggregation module by interpreting each frame-level feature as a node of a graph. The inter-relationship between all possible pairs of features, typically exploited indirectly, can be directly modeled using a graph. The module comprises a graph attention layer and a graph pooling layer followed by a readout operation. The graph attention layer first models the non-Euclidean data manifold between different nodes. Then, the graph pooling layer discards less informative nodes considering the significance of the nodes. Finally, the readout operation combines the remaining nodes into a single representation. We employ two recent systems, SE-ResNet and RawNet2, with different input features and architectures and demonstrate that the proposed feature aggregation module consistently shows a relative improvement over 10%, compared to the baseline.

Index Terms— speaker verification, feature aggregation, attention, graph attention networks, deep learning

1. INTRODUCTION

Speaker verification (SV) can be used for various authentication scenarios where the identity of a given speech is compared to that of the claimed speaker. In SV, speaker representations are derived by first extracting the frame-level features and then aggregating them. The network extracting the frame-level features is referred to as trunk network (e.g., convolutional neural network (CNN) and x-vector [1–6]). After extracting the frame-level features, various techniques including gated recurrent network (GRU), learnable dictionary encoding (LDE) are used on top of the trunk network to aggregate frame-level features into a single utterance-level feature [7–18]. The condensed utterance-level representation ultimately represents the entire character of an utterance, unlike speech recognition where each frame-level representation is of worth. Hence, feature aggregation plays an essential role and a number of studies in the recent literature have focused on developing feature aggregation methods.

A group of studies employed recurrent layers for feature aggregation. These studies modeled the frame-level representations in sequential order [7–11]. Another group of research integrated frame-level features using LDE [16–18]. Similar to the recurrent layers, the LDE layer aggregates features over time whereas it further utilizes statistics. Sequential modeling, a foundation of the above two approaches, has been used in SV based on its success in language processing and speech recognition, where the sequential order could

be a prior knowledge. However, recent studies found that sequential information may not be the key, especially in text-independent SV (TI-SV) [14, 19].

Other groups of researches exploited attention-based approaches [12–15], which exclusively emphasize important features regardless of their sequence and have recently become the mainstream in TI-SV. These approaches assign an attention weight in the form of a scalar to each frame-level feature and then perform weighted summation (also with statistics as in [14]). Each attention weight for a feature is derived via a dot product between a feature and a projection vector. A softmax non-linearity is applied to attention weights to exclusively emphasize which frame is more important, however, they are compared indirectly without modeling their relationships.

The field of graph neural networks is recently getting attention by incorporating the advantages of graph structures and deep neural networks [20, 21]. Interpreting high-dimensional representations as nodes of a graph, graph neural networks can model the non-Euclidean data manifold within nodes, including *inter-node relationship*. Particularly, recent architectures such as graph convolutional network [22] and graph attention network (GAT) [23] combined with various graph pooling layers [24–26] have demonstrated competitive performance even in image and audio domains [27–29].

Inspired by the recent success of graph neural networks, we argue that the feature aggregation could be further improved by explicitly comparing and modeling the inter-relationship between frame-level features. The feature aggregation is a task to express characteristics relative to other features and condense them into a single representation. Hence, the inter-relationship between features extends the ability of its expressiveness and could be the vital information itself. To this end, we propose a novel feature aggregation module to model inter-relationship by interpreting each frame-level feature as a node of a graph. The proposed module leverages the correlation of all possible pairs of nodes to obtain an utterance-level representation. In other words, entire frame-level features are aggregated considering their inter-relationships in addition to their intrinsic characteristics by utilizing a graph. Specifically, the aggregation involves a GAT, a graph pooling layer, and a readout operation [23, 24]. The GAT assigns different weights to each node pair (edge) and models their relationships. Then, the graph pooling layer discards less informative nodes. Finally, a single utterance-level representation is derived using the readout operation.

The effectiveness of the proposed feature aggregation framework is validated using two strong baselines, SE-ResNet [30] and RawNet2 [11], comprising different input features and architectures. Experimental results on the VoxCeleb datasets [31, 32] demonstrate more than 10% consistent improvement over the corresponding baseline. Using the proposed graph aggregation module, we could compose more lightweight, yet better performing models compared to self-attentive pooling and GRU-based aggregations.
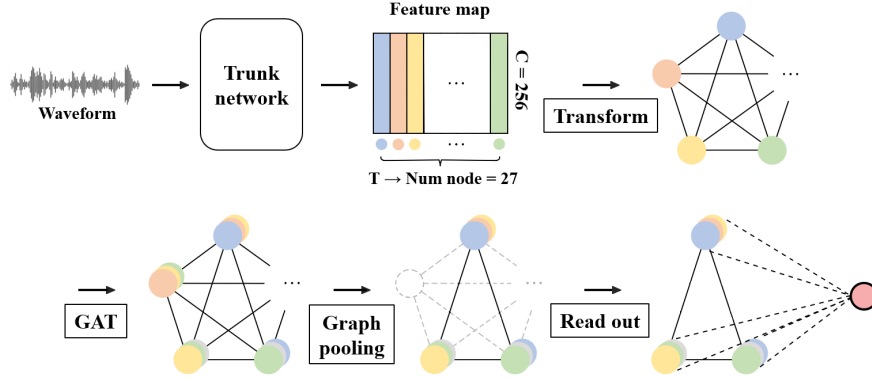

ICASSP 2022

Fig. 1. Proposed graph attentive feature aggregation module. After extracting frame-level features using a trunk network, a graph is constructed using each sequence element as a node. Each node can have a dimensionality equal to either *the number of filters* (raw waveform, 1D-CNN) or *the number of filters × the number of frequency bins* (mel-filterbank, 2D-CNN). Graph attention network first aggregates nodes features, and then the graph pooling reduces the number of nodes. The readout function integrates nodes into a single representation (best viewed in color).

2. GRAPH ATTENTIVE FEATURE AGGREGATION

In this section, we introduce the graph attentive feature aggregation module. The proposed module is located after the extraction of frame-level features using a trunk network (RawNet2 or SE-ResNet). The proposed approach consists of three components: i) graph attention layer, ii) graph pooling layer, and iii) readout. We first describe how we formulate a graph from a feature map and perform GAT to integrate the overall features. Then, we explain how the graph pooling layer discards less informative nodes and the readout combines the results into a single representative feature. The overall scheme is illustrated in Figure 1 using RawNet2 (the scheme is identical for SE-ResNet except for the dimensionality of each node).

2.1. Graph attention layer

We first formulate a graph using frame-level features extracted from a trunk network. Specifically, we interpret each sequence element as a node of a graph; for RawNet2, each node has a dimensionality equal to *the number of filters* and for SE-ResNet, the dimensionality equals to *the number of filters × the number of frequency bins*. Note that we design a bi-directional graph where there is a bi-directional edge from every node to another node. Nodes can be directly compared considering all combinations/relationships because all nodes have bi-directional edges. Hence, entire frame-level features are aggregated considering their inter-relationships as well as intrinsic characteristics of their own by utilizing the graph attention layer.

The GAT process can be described as follows:

$$\mathbf{n}' = \mathbf{x}W, \tag{1}$$

$$e_{ij} = LeakyReLU(\gamma \cdot \mathsf{concat}(n'_i, n'_j)), \tag{2}$$

$$a_{ij} = \mathsf{softmax}(e_{ij}) = \frac{\mathsf{exp}(e_{ij})}{\sum_{t=1}^{N} \mathsf{exp}(e_{it})}, \tag{3}$$

$$n_i = \sum_{j=1}^{N} a_{ij} n'_j. \tag{4}$$

Let $\mathbb{G}$ be a bi-directional graph comprising N nodes with the F dimensional features of each node. A set of nodes in $\mathbb{G}$ is defined as $\mathbf{x} \in \mathbb{R}^{N \times F}$ and each nodes are represented as row vectors

x_1, x_2, $\cdots$, x_N. In the GAT layer, $\mathbf{x}$ is first projected into F' dimensional space using a matrix multiplication with $W \in \mathbb{R}^{F \times F'}$ resulting in $\mathbf{n}' \in \mathbb{R}^{N \times F'}$ (Equation (1)). Then, we calculate attention scores where an attention score a_{ij} between i-th node n'_i and j-th node n'_j ($n'_i, n'_j \in \mathbb{R}^{F'}$, $i \neq j$) is calculated through Equation (2) and (3). We derive e_{ij}, by first concatenating two nodes n'_i and n'_j and then projecting to a scalar value through a dot product with $\gamma \in \mathbb{R}^{2F' \times 1}$, a learnable parameter, followed by a Leaky ReLU non-linearity [33]. Using the calculated attention scores, the GAT performs self-attention on node n'_i as in Equation (4). The GAT is applied to every node $\mathbf{x}$ transforming $\mathbf{n}'$ to $\mathbf{n}$ and then the output $\mathbf{n}$ is fed to a graph pooling layer.

2.2. Graph pooling layer and readout

We utilize a graph pooling layer followed by the readout operation to obtain an utterance-level feature. Similar to pooling layers in convolutional neural networks, graph pooling can reduce the size of features to enable high-level feature encoding and receptive field enlargement. Specifically, the graph pooling layer reduces the original graph into a sub-graph by removing less informative nodes as depicted in Figure 1. For the graph pooling, we exploit gPool introduced in the Graph U-Net architecture [24]. There are various other graph pooling methods, but many of them assume the hierarchical architecture of the graph. As it is difficult to define a hierarchical relationship between frame-level features in audio data, we adopt gPool, which does not require the graph to have a hierarchical structure. gPool consists of the following processes: projection, top-K node selection, and a gating mechanism.

Formally, gPool can be denoted as follows:

$$
\begin{aligned}
\mathbf{y} &= \mathbf{n}^l \mathbf{p}^l / \|\mathbf{p}^l\|, \\
\mathrm{index} &= \mathsf{rank}(\mathbf{y}, K), \\
\tilde{\mathbf{y}} &= \mathsf{sigmoid}(\mathbf{y}(\mathrm{index})), \\
\tilde{\mathbf{n}}^l &= \mathbf{n}^l(\mathrm{index}, :) \\
\mathbf{n}^{l+1} &= \tilde{\mathbf{n}}^l \odot \tilde{\mathbf{y}}
\end{aligned}
\tag{5}
$$

A set of nodes in the graph pooling layer l is represented as $\mathbf{n}^l$. Note that the adjacency matrix is omitted because we utilize a bi-

Table 1. SE-ResNet architecture applying the proposed graph aggregation method. ReLU and batch normalization layers are omitted. Squeeze and excitation [34] is performed after all convolutional blocks. The numbers denoted in brackets refer to **[size× size, filters]** and those in the output refer to **(filters, frequency, time)**. The output of the last fully-connected layer is used as the speaker embedding.

layer name	Filters	Output
conv1	3×3, 32, stride 1	(32×40×T)
conv2	$\begin{bmatrix} 3×3, 32 \\ 3×3, 32 \end{bmatrix}$ × 3, stride 1	(32×40×T)
conv3	$\begin{bmatrix} 3×3, 64 \\ 3×3, 64 \end{bmatrix}$ × 4, stride 2	(64×20×T/2)
conv4	$\begin{bmatrix} 3×3, 128 \\ 3×3, 128 \end{bmatrix}$ × 6, stride 2	(128×10×T/4)
conv5	$\begin{bmatrix} 3×3, 128 \\ 3×3, 128 \end{bmatrix}$ × 3, stride 2	(128 × 5 ×T/8)
aggregation	GAT	(640)
fc	FCN, 256	(256)

Table 2. Application of proposed graph module to SE-ResNet and RawNet2. SAP in SE-ResNet and GRU in RawNet2 are both replaced by GAT.

Feature extractor	Aggregation	# Params	EER (%)
SE-ResNet	SAP	6.0M	1.98
SE-ResNet	GAT	5.4M	**1.86**
RawNet2	GRU	13.2M	2.48
RawNet2	GAT	9.9M	**2.23**

directional graph that is fully connected and different from the original paper of [24]. In the projection stage, gPool obtains score vector y by estimating the scalar projection values of each node, using a trainable projection vector $p \in \mathbb{R}^{F'}$. In the top-K node selection, the operation of node ranking (rank) outputs indices of the K-largest values (index) based on calculated score vector y. The score vector of selected nodes, $\tilde{y}$, can be obtained by applying sigmoid to each element in the extracted scalar projection vector. In the gate stage, element-wise product $\odot$ is conducted between the pooled nodes $\tilde{n}^l$ and the selected node scores $\tilde{y}$.

After the graph pooling, we conduct the readout operation, which combines the processed nodes into a single node. We explore various types of readout operation where the best result is obtained using the summation of nodes:

$$U = \sum_{k=1}^{K} n_k^{l+1}. \tag{6}$$

Here, U is the utterance-level feature derived by aggregating the K nodes n_k^{l+1}. The results of various readout mechanisms can be found in Section 4.

3. EXPERIMENTS

3.1. Datasets

All experiments are performed using the VoxCeleb1&2 datasets [31, 32]. We train the model using the development subset of VoxCeleb2

Table 3. Comparison of applying different pooling ratios in graph pooling layer. A higher pooling ratio means more nodes remain.

Pooling method	EER (%)	
	SE-ResNet	RawNet2
w/o gPool	1.86	2.23
Top 11% gPool	2.00	2.39
Top 33% gPool	1.91	2.17
Top 80% gPool	**1.75**	**2.15**

Table 4. Comparison of various readout methods. "Combine by concat" indicates that the results of summation, standard deviation (std), minimum, and maximum are concatenated by reducing the output dimensions in one-quarter.

Operation	EER (%)
Mean	2.38
Sum	**2.23**
Max	2.3
Combine by concat	2.38

that includes the utterances from 5,994 speakers. Then, the evaluation is performed using the original trial that uses VoxCeleb1's evaluation subset.

3.2. Implementation details

Both baselines using a mel-filterbank and a raw waveform are implemented based on the PyTorch framework [38]. Graphs are implemented using Deep Graph Library [39]. We adjust the number of attention heads and use 16 heads in RawNet2 and 32 heads in SE-ResNet based on empirical results.

SE-ResNet. Our mel-filterbank baseline is most similar to the architecture of [30], but overall details are adjusted. We use 40-dimensional mel-filterbank features extracted with 1,024 point FFT and a hamming window of width 25ms and step 10ms. The mel-filterbank baseline is optimized using Adam optimizer which uses a learning rate of 0.001, and the learning rate is decreased by 5% in each epoch. We use two types of loss functions, additive angular margin softmax (AAM-softmax) [40] and angular prototypical loss (AP) [41] to consider inter-class as well as intra-class covariance. We employ a margin of 0.3 and a scale of 30 for AAM-softmax. For AP, we use a mini-batch size of 200, where each mini-batch contains two utterances per speaker. The system is trained for 100 epochs. Our overall architecture applying the proposed method is presented in Table 1.

RawNet2. RawNet2 [11] is an end-to-end system that is fed by raw waveforms directly without preprocessing techniques. For mini-batch construction, utterances are either cropped or duplicated (concatenated) into 59,049 samples ($\approx$ 3.69s) in the training phase, following [11]. In the evaluation phase, no adjustments are made to the length. We modify several details from RawNet2 [11] in the process of adjusting the GAT as follows: i) exclude sinc-conv layer, as it slows down training time in spite of showing similar performance, ii) replace softmax with additive margin softmax (AM-softmax) [42], iii) reduce the dimensions by half for the last fully-connected layer before speaker embedding (1024 dimensions to 512 dimensions).

Table 5. Results of comparison with recent state-of-the-art systems. The two baselines of this study are each compared with various architectures that use the same input feature.

	Input Feature	Front-end	Aggregation	Loss	EER (%)
Chung *et al.* [2]	Spec-257	Thin ResNet-34	SAP	AP	2.21
Yu *et al.* [35]	Spec-512	ResNet-50	TAP	EAM-Softmax	2.94
Liu *et al.* [36]	MFB-40	Dense-Residual	ABP	Softmax	2.54
Jung *et al.* [1]	MFB-40	Fast ResNet-34	LDE	NP + Softmax	1.98
Kye *et al.* [15]	MFB-40	Fast ResNet-34	CAP	NP + Softmax	1.88
Ours	MFB-40	SE-ResNet	SAP	AP + AAM-Softmax	1.98
Ours(Proposed)	MFB-40	SE-ResNet	GAT	AP + AAM-Softmax	**1.75**
Lin *et al.* [3]		wav2spk	Gating + SP	AM-Softmax	3.00
Zhu *et al.* [37]	Raw waveform	Y-vector	SP	AM-Softmax	2.60
Jung *et al.* [11]		RawNet2	GRU	Softmax	2.48
Ours(Proposed)		RawNet2	GAT	AM-Softmax	**2.15**

Table 6. Results of additional experiments. The first row shows the result of applying GAT and GRU sequentially. The lower two rows show the result of applying two GAT layers with different pooling architectures.

System	EER (%)
GAT - GRU	4.16
2GAT - Global architecture	2.66
2GAT - Hierarchical architecture	2.55

4. RESULTS

Application of GAT. In Table 2, we first address the effectiveness of the graph attention layer on our two baselines: SE-ResNet and RawNet2. Here, the graph pooling is temporarily excluded and summation readout operation is applied. Adoption of the graph attention layer reduced the EER relatively by 6% and 10%, respectively. In addition, we confirm that the number of parameters also decreases (by almost 25% in the case of RawNet2 (13.2M to 9.9M)).

Graph pooling. In Table 3, we explore the effect of applying the graph pooling layer between the graph attention layer and the readout operation with diverse pooling ratios. The denoted percentage of pooling refers to the remaining ratios of nodes; higher ratios mean that fewer nodes have been removed. Top 80% pooling removing 20% of irrelevant nodes shows the best results where the relative improvement compared to the baseline becomes 11.6% and 13.3%. Through a comparison of the three different ratios of pooling, we find that discarding too many nodes worsens the performance (11% pool) but removing a few of the most irrelevant nodes improves the performance (80% pool). We thus conclude that the discriminative information is well combined into the majority of nodes (80% in our case) via the GAT layer.

Comparison of readout operations. In Table 4, we further compare several readout operations using the RawNet2 baseline. We find that the proposed graph-based aggregation shows consistent effectiveness, regardless of the adopted readout approach because all node aggregation methods performed better than GRU. Among various methods, summation achieved the best result.

Comparison with state-of-the-art systems. In Table 5, we compare the proposed system with other recent state-of-the-art systems in the literature that adopt various feature aggregation methods. None of the systems apply data augmentation techniques. In both input fea-

tures, the proposed models using the proposed graph aggregation module demonstrated the best performance, showing the effectiveness of the proposed approach. Given that we did not use data augmentation methods, there is room to improve the system using various data augmentation.

Additional experiments. Table 6 addresses two additional experiments using the RawNet2 baseline. First, we explore whether adopting a GRU after the GAT is beneficial and show the result in the first row. To maintain the overall complexity of the model, we omitted the last residual block and then placed the GAT and the GRU in sequence. However, this worsens the performance, as presented in the first row. In our analysis, this shows that applying GRU directly to the GAT's output is not effective, because the GAT's output is not sequential.

Second, we explore using two GAT layers with different pooling architectures proposed in [25] and denote the result in second and third rows. Both architectures adopt two GAT layers in sequence. The global architecture concatenates two GAT layers' output and feeds it to the graph pooling layer followed by the readout, whereas hierarchical readout performs graph pooling and readout after each GAT layer and then adds them element-wisely. Through experiments, we found that both modified pooling architectures did not bring further improvements.

5. CONCLUSIONS

In this paper, we proposed a graph feature aggregation method for TI-SV. Utilizing a GAT, graph pooling layer, and readout operation, we directly modeled the inter-relationship between entire frame-level features, which is partially or indirectly utilized in the existing methods. As this is the first work employing graph neural networks for feature aggregation, we also explored various configurations to optimize the system. Consistent improvements over the baselines with different aggregation modules demonstrate the effectiveness of the proposed approach.

6. REFERENCES

[1] Y. Jung, S. M. Kye, Y. Choi et al., "Improving multi-scale aggregation using feature pyramid module for robust speaker verification of variable-duration utterances," *in Proc. Interspeech*, 2020.

[2] J. S. Chung, J. Huh, S. Mun et al., "In defence of metric learning for speaker recognition," in *in Proc. Interspeech*, 2020.

[3] W.-W. Lin and M.-W. Mak, "Wav2spk: A simple dnn architecture for learning speaker embeddings from waveforms.," in *in Proc. Interspeech*, 2020.

[4] D. Snyder, D. Garcia-Romero, G. Sell et al., "X-vectors: Robust dnn embeddings for speaker recognition," in *Proc. ICASSP*. IEEE, 2018.

[5] Y. Tang, G. Ding, J. Huang et al., "Deep speaker embedding learning with multi-level pooling for text-independent speaker verification," in *Proc. ICASSP*. IEEE, 2019.

[6] Y. Wu, C. Guo, H. Gao et al., "Vector-based attentive pooling for text-independent speaker verification.," in *in Proc. Interspeech*, 2020.

[7] C. Li, X. Ma, B. Jiang et al., "Deep speaker: an end-to-end neural speaker embedding system," *arXiv preprint arXiv:1705.02304*, 2017.

[8] J.-w. Jung, H.-S. Heo, H.-j. Shim and H.-J. Yu, "Short utterance compensation in speaker verification via cosine-based teacher-student learning of speaker embeddings," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2019.

[9] Y. Zhang, M. Yu, N. Li et al., "Seq2seq attentional siamese neural networks for text-dependent speaker verification," in *Proc. ICASSP*. IEEE, 2019.

[10] J.-w. Jung, H.-S. Heo, J.-h. Kim et al., "Rawnet: Advanced end-to-end deep neural network using raw waveforms for text-independent speaker verification," *in Proc. Interspeech*, 2019.

[11] J.-w. Jung, S.-b. Kim, H.-j. Shim et al., "Improved rawnet with feature map scaling for text-independent speaker verification using raw waveforms," *in Proc. Interspeech*, 2020.

[12] D. Snyder, D. Garcia-Romero, D. Povey and S. Khudanpur, "Deep neural network embeddings for text-independent speaker verification.," in *in Proc. Interspeech*, 2017.

[13] Y. Zhu, T. Ko, D. Snyder et al., "Self-attentive speaker embeddings for text-independent speaker verification.," in *in Proc. Interspeech*, 2018.

[14] K. Okabe, T. Koshinaka and K. Shinoda, "Attentive statistics pooling for deep speaker embedding," *in Proc. Interspeech*, 2018.

[15] S. M. Kye, Y. Kwon and J. S. Chung, "Cross attentive pooling for speaker verification," in *2021 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2021.

[16] W. Cai, J. Chen and M. Li, "Exploring the encoding layer and loss function in end-to-end speaker and language recognition system," *in Proc. Interspeech*, 2018.

[17] W. Xie, A. Nagrani, J. S. Chung and A. Zisserman, "Utterance-level aggregation for speaker recognition in the wild," in *Proc. ICASSP*. IEEE, 2019.

[18] Y. Jung, Y. Kim, H. Lim et al., "Spatial pyramid encoding with convex length normalization for text-independent speaker verification," *in Proc. Interspeech*, 2019.

[19] B. Desplanques, J. Thienpondt and K. Demuynck, "Ecapa-tdnn: Emphasized channel attention, propagation and aggregation in tdnn based speaker verification," *in Proc. Interspeech*, 2020.

[20] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *Knowledge-Based Systems*, 2018.

[21] Z. Wu, S. Pan, F. Chen et al., "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, 2020.

[22] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[23] P. Veličković, G. Cucurull, A. Casanova et al., "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[24] H. Gao and S. Ji, "Graph u-nets," in *Proc. ICML*, 2019.

[25] J. Lee, I. Lee and J. Kang, "Self-attention graph pooling," in *Proc. ICML*, 2019.

[26] H. Yuan and S. Ji, "Structpool: Structured graph pooling via conditional random fields," in *Proc. ICLR*, 2020.

[27] J.-w. Jung, H.-S. Heo, H.-J. Yu and J. S. Chung, "Graph attention networks for speaker verification," in *Proc. ICASSP*. IEEE, 2021.

[28] H. Tak, J.-w. Jung, J. Patino et al., "Graph attention networks for anti-spoofing," *in Proc. Interspeech*, 2021.

[29] H. Tak, J.-w. Jung, J. Patino et al., "End-to-end spectro-temporal graph attention networks for speaker verification anti-spoofing and speech deepfake detection," *arXiv preprint arXiv:2107.12710*, 2021.

[30] H. S. Heo, B.-J. Lee, J. Huh and J. S. Chung, "Clova baseline system for the VoxCeleb speaker recognition challenge 2020," *arXiv preprint arXiv:2009.14153*, 2020.

[31] A. Nagrani, J. S. Chung and A. Zisserman, "Voxceleb: a large-scale speaker identification dataset," *in Proc. Interspeech*, 2017.

[32] J. S. Chung, A. Nagrani and A. Zisserman, "Voxceleb2: Deep speaker recognition," *in Proc. Interspeech*, 2018.

[33] A. L. Maas, A. Y. Hannun, A. Y. Ng et al., "Rectifier non-linearities improve neural network acoustic models," in *Proc. ICML*, 2013.

[34] J. Hu, L. Shen and G. Sun, "Squeeze-and-excitation networks," in *Proc. CVPR*, 2018.

[35] Y.-Q. Yu, L. Fan and W.-J. Li, "Ensemble additive margin softmax for speaker verification," in *Proc. ICASSP*. IEEE, 2019, pp. 6046–6050.

[36] Y. Liu, Y. Song, I. McLoughlin et al., "An effective deep embedding learning method based on dense-residual networks for speaker verification," in *Proc. ICASSP*. IEEE, 2021.

[37] G. Zhu, F. Jiang and Z. Duan, "Y-vector: Multiscale waveform encoder for speaker embedding," in *in Proc. Interspeech*, 2021.

[38] A. Paszke, S. Gross, F. Massa et al., "Pytorch: An imperative style, high-performance deep learning library," *NIPS*, 2019.

[39] M. Wang, L. Yu, D. Zheng et al., "Deep graph library: Towards efficient and scalable deep learning on graphs.," 2019.

[40] J. Deng, J. Guo, N. Xue and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *Proc. CVPR*, 2019.

[41] J. Wang, F. Zhou, S. Wen et al., "Deep metric learning with angular loss," in *Proc. ICCV*, 2017.

[42] H. Wang, Y. Wang, Z. Zhou et al., "Cosface: Large margin cosine loss for deep face recognition," in *Proc. CVPR*, 2018.