GRAPH ATTENTIVE FEATURE AGGREGATION FOR TEXT-INDEPENDENT SPEAKER VERIFICATION

Hye-jin Shim†, Jungwoo Heo†, Jae-han Park‡, Ga-hui Lee‡, and Ha-Jin Yu†

†School of Computer Science, University of Seoul, ‡KT Corporation

ABSTRACT

The objective of this paper is to combine multiple frame-level features into a single utterance-level representation considering pairwise relationship. For this purpose, we propose a novel graph attention 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 a 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 pairs (edges) 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.
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 function integrates nodes into a single representation (best viewed in color).

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 \( \times \) the number of frequency bins. Note that we design a fully-connected (complete) graph where all possible edges exist between pairs of nodes. 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.

Let \( G \) be a complete graph comprising \( N \) nodes with the \( F \) dimensional features of each node. A set of nodes in \( G \) is defined as \( x \in \mathbb{R}^{N \times F} \) and each nodes are represented as row vectors \( x_1, x_2, \ldots, x_N \). In the GAT layer, \( x \) 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 \( x \) transforming \( \mathbf{n}' \) to \( \mathbf{n} \) and then the output \( \mathbf{n} \) is fed to a graph pooling layer.

The GAT process can be described as follows:

\[
\mathbf{n}' = xW, \quad (1)
\]

\[
e_{ij} = \text{LeakyReLU}(\gamma \cdot \text{concat}(n'_i, n'_j)), \quad \text{(2)}
\]

\[
a_{ij} = \text{softmax}(e_{ij}) \frac{\exp(e_{ij})}{\sum_{t=1}^{N} \exp(e_{it})}, \quad \text{(3)}
\]

\[
n_i = \sum_{j=1}^{N} a_{ij} n'_j, \quad \text{(4)}
\]

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
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    |

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

| Pooling method | EER (%) |
|---------------|---------|
| w/o gPool     | 1.86    |
| Top 11% gPool | 2.00    |
| Top 33% gPool | 1.91    |
| Top 80% gPool | 1.75    |

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   |

Here, \( U \) is the aggregated utterance-level feature and \( K \) nodes \( n_k \) are integrated to \( n \). 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 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 (≈ 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).
The systems apply data augmentation techniques. In both input features, various methods demonstrate 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.

### Acknowledgement

We would like to thank Jee-weon Jung at Naver Corporation for his help with the conceptualization and editing.

### 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.
