Compact Graph Architecture for Speech Emotion Recognition

Amir Shirian, Tanaya Guha
University of Warwick

Abstract
We propose a deep graph approach to address the task of speech emotion recognition. A compact, efficient and scalable way to represent data is in the form of graphs. Following the theory of graph signal processing, we propose to model speech signal as a cycle graph or a line graph. Such graph structure enables us to construct a graph convolution network (GCN)-based architecture that can perform an accurate graph convolution in contrast to the approximate convolution used in standard GCNs. We evaluated the performance of our model for speech emotion recognition on the popular IEMOCAP database. Our model outperforms standard GCN and other relevant deep graph architectures indicating the effectiveness of our approach. When compared with existing speech emotion recognition methods, our model achieves state-of-the-art performance (4-class, 65.29%) with significantly fewer learnable parameters.

Index Terms: Speech Emotion recognition, Graph convolutional networks, Graph signal processing.

1. Introduction

Machine recognition of emotional content in speech is crucial in many human-centric systems, such as behavioral health monitoring [1] and empathetic conversational systems [2]. Speech emotion recognition (SER) is a challenging task due to the huge variability in emotion expression and perception across speakers, languages and cultures.

The majority of approaches in the SER literature follows a two-stage approach. First, a set of low-level descriptors (LLDs) are extracted from raw audio. The LLDs are then input to a deep learning model to generate discrete emotion labels [3][4][5][6]. While extracting hand-crafted acoustic features is more common, lexical features have been also shown to be useful [7][8]. Convolutional neural networks (CNNs) have been used with log Mel spectrograms as input to learn features [9], but without explicitly considering the temporal dynamics in speech. Explicit modeling of the temporal dynamics is important as it reflects the changes in emotion dynamics [10]. To capture the temporal dynamics, recurrent neural networks (RNNs) [5][9] including their attention-based long-short term memory network (LSTMs) variants are a common choice [4][5]. The RNNs and LSTMs, predominant in SER, often lead to complex architecture with millions of trainable parameters.

A compact, efficient and scalable way to represent data is in the form of graphs. In the last few years, graph convolutional networks (GCNs) [11] have been successfully used to address various problems in computer vision and natural language processing, such as action recognition [12], tracking [13] and text classification [14]. In the area of audio processing, a recent work has proposed an attentional graph neural network (GNN) to address the problem of few-shot audio classification [15]. The authors are not aware of any other GNN or GCN-based work in audio analysis.

Motivated by the success of GCNs, we propose to adopt a deep graph approach to SER. We base our work on spectral GCNs which have a strong foundation on graph signal processing [16]. Spectral GCNs perform convolution operation on the spectrum of the graph Laplacian considering the convolution kernel (diagonal matrix) to be learnable [17]. This involves eigen decomposition of the graph Laplacian matrix, which is computationally expensive. To reduce the computational cost, ChebNet approximates the convolution operation (including the learnable convolution kernel) in terms of Chebyshev polynomials and [18]. The most popular form of GCN uses a first order approximation of the Chebyshev polynomial to further simplify the convolution operation to a linear projection [11]. This GCN model is simple to implement, and has been successfully used for various node classification tasks in social media networks and citation networks.

In this paper, we cast SER as a graph classification problem. We model a speech signal as a graph, where each node corresponds to a short windowed segment of the signal. Each node is connected to only two adjacent nodes thus transforming the signal to a line graph or a cycle graph. Owing to this particular graph structure, we take advantage of results in graph signal processing [19] to perform accurate graph convolution (in contrast to the approximations used in popular GCNs [11]). This leads to a light-weight GCN-based model with superior emotion recognition performance on the IEMOCAP database [20]. To summarize, our contributions are as follows:

(i) To the best of our knowledge, this is the first work that takes a graph classification approach to SER.

(ii) Leveraging theories from graph signal processing, we propose a GCN-based graph classification approach that can efficiently perform accurate graph convolution.

(iii) Our model, despite having smaller size, achieves superior performance on the IEMOCAP database outperforming relevant and competitive baselines.

2. Proposed Approach

In this section, we describe our graph classification approach to SER. First, we transform each speech sample to a graph. Next, we propose our GCN architecture that classifies each graph according to the emotion label. Fig. 1 gives an overview of our approach. Below, we describe each component in detail.

2.1. Graph Construction

Given a speech signal (utterance), the first step is to construct a corresponding graph \( G = (V, E) \), where \( V \in \{v_i\}_{i=1}^M \) is the set of \( M \) nodes, and \( E \) is the set of all edges between the nodes. The adjacency matrix of \( G \) is \( A \in \mathbb{R}^{M \times M} \), where an element \( (A)_{ij} \) denotes the edge weight connecting \( v_i \) and \( v_j \).

Our graph construction strategy follows a simple frame-to-node transformation, where \( M \) frames (short, overlapping seg-
Figure 1: Our proposed graph-based architecture for SER consists of two graph convolution layers and a pooling layer to learn graph embedding from node embeddings to facilitate emotion classification.

Figures 2: Graph construction from speech input. (a) LLDs are extracted as node features \( x_i \) from raw speech segments. (b) cycle graph, and (c) chain graph.

The two graph structures are significant because of the special structures of their graph Laplacians, which significantly simplifies spectral GCN operation. This is discussed in the following section in more detail.

Each node \( v_i \) is also associated with a node feature vector \( x_i \in \mathbb{R}^P \). The node feature vectors contain acoustic features extracted from the corresponding speech segment. A feature matrix \( X \in \mathbb{R}^{M \times P} \) containing all node feature vectors is defined as \( X = [x_1, \ldots, x_M] \).

2.2. Graph Classification

Given a set of (utterances transformed to) graphs \( \{G_1, ..., G_N\} \) and their true labels \( \{y_1, ..., y_N\} \) represented as one-hot vectors, our task is to develop a GCN architecture that is able to recognize the emotional content in the utterances. Fig. 4 presents an overview of our architecture comprising two graph convolution layers, a pooling layer that yields a graph-level embedding vector, followed by a fully connected layer that produces the classification labels.

Graph convolution layer. We base our model on a spectral GCN, which performs graph convolution in the spectral domain. Following the theory of graph signal processing \cite{Shuman2013}, graph convolution in time domain is defined as

\[
    h = x_i \ast w
\]

where \( w \) is the graph convolution kernel (learnable) and \( x_i \) is the input node features (for simplicity, consider each node with a single value). This is equivalent to a product in the graph spectral domain.

\[
    \hat{h} = \hat{x}_i \ast \hat{w}
\]

where \( \hat{h} \), \( \hat{x}_i \), and \( \hat{w} \) denote the filtered output, node features and the convolution filter in the graph spectral domain, respectively. Adopting a matrix notation and a node feature matrix, we have

\[
    \hat{H} = \hat{X}W
\]

In order to have \( \hat{X} \) and \( \hat{W} \), we compute the normalized graph Laplacian matrix

\[
    \mathcal{L} = D^{-\frac{1}{2}}LD^{-\frac{1}{2}}
\]
where $D$ is degree matrix and $L = D - A$ with $A$ being the adjacency matrix of the graph. The degree matrix $D$ is a diagonal matrix where the $i$th diagonal element denotes the degree of $v_i$ given by $\deg(v_i) = \sum_j A_{ij}$.

The eigen decomposition of $L$ can be written as

$$L = U \Lambda U^T = \sum_{i=1}^{M} \lambda_i u_i u_i^T$$

(3)

where $\lambda_i$ is the $i$th eigen value of $L$ corresponding to the eigen vector $u_i$. $\Lambda = \text{diag}(\lambda_i)$ and $U = [u_1, u_2 \cdots u_N]$. The exact graph convolution operation is thus defined as

$$\tilde{H} = (U^T X)(U^T W)$$

$$H = U \tilde{H}$$

The graph convolution propagation at $k$th layer thus becomes

$$H^{(k+1)} = U \left( U^T H^{(k)} (U^T W^{(k)}) \right)$$

(4)

where $H^{(0)} = X$ and $W$ is learnable. Not that for $A = A_c$ (cycle graph), $L$ takes the following form

$$L = \begin{bmatrix}
2 & -1 & 0 & \cdots & -1 \\
-1 & 2 & -1 & \cdots & 0 \\
0 & -1 & 2 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
-1 & 0 & \cdots & -1 & 2
\end{bmatrix}$$

The $L$ is circulant and GFT is equivalent to discrete Fourier transform (DFT) [19]. Similarly, for $A = A_l$ (line graph), GFT is equivalent to discrete cosine transform (DCT). This makes the convolution operation convenient and computationally efficient as we can avoid eigen decomposition (can be computationally expensive for arbitrary graph).

Following a recent work on spatial GCN [21], we propose to learn the convolution kernel Eq. [5] in terms of a a multi-layer perceptron (MLP). Finally, our convolution operation takes the following form

$$H^{(k+1)} = U \left( \text{MLP}(U^T H^{(k)}) \right)$$

(5)

where, only the MLP parameters are learnable.

**Pooling layer.** Our objective is to classify entire graphs. Hence, we need a function to attain a graph-level representation $h_G \in \mathbb{R}^d$ from the node-level embeddings. This can be obtained by pooling the node-level embeddings $H^{(k)}$ at the final layer $k = K$ before passing them on to the classification layer. Common choices for pooling functions in graph domain are mean, max and sum pooling [11][22]. Max and mean pooling often fail to preserve the underlying information about the graph structure while sum pooling has shown to be a better alternative [21]. We use sum pooling to obtain the graph-level representation:

$$h_G = \text{sumpool}(H^{(K)}) = \sum_{i=1}^{M} h_i^{(K)}$$

(6)

The pooling layer is followed by one fully-connected layer which produces the classification labels. Our GCN-based model is trained with the cross-entropy loss $= -\sum y_n \log \hat{y}_n$.

### Table 1: SER results and comparison on the IEMOCAP databases in terms of weighted accuracy (WA) and unweighted accuracy (UA).

<table>
<thead>
<tr>
<th>Model</th>
<th>WA (%)</th>
<th>UA (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Graph baselines</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PATCHY-SAN [22]</td>
<td>60.34</td>
<td>56.27</td>
</tr>
<tr>
<td>PATCHY-Diff [26]</td>
<td>63.23</td>
<td>58.71</td>
</tr>
<tr>
<td><strong>SER models</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attn-BlSTM 2016 [5]</td>
<td>59.33</td>
<td>49.96</td>
</tr>
<tr>
<td>BLR 2017 [27]</td>
<td>62.54</td>
<td>57.85</td>
</tr>
<tr>
<td>RNN 2017 [6]</td>
<td>63.50</td>
<td>58.80</td>
</tr>
<tr>
<td>CRNN 2018 [28]</td>
<td>63.98</td>
<td>60.35</td>
</tr>
<tr>
<td>SegCNN 2019 [9]</td>
<td>64.53</td>
<td>62.34</td>
</tr>
<tr>
<td>CNN 2019 [29]</td>
<td>58.52</td>
<td>-</td>
</tr>
<tr>
<td>LSTM 2019 [29]</td>
<td>58.72</td>
<td>-</td>
</tr>
<tr>
<td>CNN-LSTM 2019 [29]</td>
<td>59.23</td>
<td>-</td>
</tr>
<tr>
<td><strong>Ours (cycle)</strong></td>
<td>65.29</td>
<td>62.27</td>
</tr>
<tr>
<td><strong>Ours (line)</strong></td>
<td>64.69</td>
<td>61.14</td>
</tr>
<tr>
<td><strong>Ours (cycle w/o MLP)</strong></td>
<td>64.19</td>
<td>60.31</td>
</tr>
</tbody>
</table>

### 3. Experiments

In this section, we present experimental results and analysis to evaluate the performance of the proposed GCN architecture.

#### 3.1. Database

We evaluated our model on the popular IEMOCAP database [20]. This database contains a total of 12 hours of data collected over 5 dyadic sessions with 10 subjects. At least three evaluators annotated each utterance with one of the six emotion labels (anger, excitement, frustration, joy, neutral and sadness). A single utterance may have multiple labels owing to different annotators. We consider only the label which has majority agreement. To be consistent with previous studies, we only used four emotion classes: anger, joy, neutral, and sadness. The final dataset contains a total of 4490 utterances including 1103 anger, 595 joy, 1708 neutral and 1084 sad.

#### 3.2. Node features

We extract a set of low-level descriptors (LLDs) from the raw speech utterances as proposed for Interspeech2009 emotion challenge [23] using the OpenSMILE toolkit [24]. The feature set includes Mel-frequency cepstral coefficients (MFCCs), zero-crossing rate, voice probability, fundamental frequency (F0) and frame energy. For each sample, we use a sliding window of length 25ms (with a stride length of 10ms) to extract the LLDs locally. Each feature is then smoothed using a moving average filter, and the smoothed version is used to compute their respective first order delta coefficients. In addition, motivated by a recent work on speech emotion recognition [25], we also add spontaneity as a binary feature. The spontaneity information comes with the database. Altogether this produces node feature vectors of dimension $P = 35$. 
Figure 3: Confusion matrix in terms of WA (%) for our model (w/ cycle graph structure).

Table 2: Model size in terms of learnable parameters

<table>
<thead>
<tr>
<th>Model</th>
<th>GCN</th>
<th>PATCHY-SAN</th>
<th>PATCHY-Diff</th>
<th>BLSTM</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>~76K</td>
<td>~60K</td>
<td>~68K</td>
<td>~0.8M</td>
<td>~30K</td>
<td></td>
</tr>
</tbody>
</table>

3.3. Implementation Details

Each audio sample produces a graph of $M = 120$ nodes, where each node corresponds to a (overlapping) speech segment of length 25ms. Padding is used to make the samples of equal length as before. We perform a 5-fold cross-validation and report both average weighted and unweighted accuracies on the IEMOCAP database.

Our network weights are initialized following the Xavier initialization [30]. We used Adam optimizer with a learning rate of 0.01 and a decay rate of 0.5 after each 50 epochs for all experiments. We used Pytorch for implementing our model and the baselines on an NVIDIA RTX-2080Ti GPU.

3.4. Results and Analysis

Comparison with graph-based models. We compare our model against three state-of-the-art deep graph models using the same node features and a cycle graph structure.

GCN [11]. A natural baseline to compare with our model is a spectral GCN in its standard form. The original network [11] is designed for node classification and only yields node-level embeddings. To obtain a graph-level embedding, we used the sum pooling function.

PATCHY-SAN [22]. A recent architecture that learns CNNs for arbitrary graphs. This architecture is originally developed for graph classification.

PATCHY-Diff [26]. A recent work on hierarchical GCN proposes to use differentiable pooling layer between graph convolution layers. We used this pooling layer with PATCHY-SAN as in the original paper.

Table 1 compares our model against these graph-based models in terms of SER accuracy. All the baseline models use the same node features as ours. Clearly, our model outperforms all the baselines by a significant margin. Compared to the popular GCN [11], our model improves the recognition accuracy by more than 9%. This result indicates that accurate convolution in graph domain improves the accuracy significantly.

Comparison with SER state-of-the-art. In addition to the graph models, we compared our model with a number of recent methods on SER: attention-based bidirectional LSTM (Attn-BLSTM) [4], Bayesian logistic regression (BLR) [27], RNN [6], statistical features with convolutional RNN (RNN) [28], SegCNN [9], CNN [29], LSTM [29], and CNN-LSTM [29]. All models except the CNN, LSTM, and CNN-LSTM use LLDs as input features. Our model outperforms all LSTM-based architectures - a class of classifier most commonly used in SER. Our model achieves highest weighted accuracy (WA) when a cycle graph structure is used outperforming all others. Our model's unweighted accuracy (UA) is the same as that of SegCNN [9], but our model has significantly fewer parameters (30K learnable parameters vs. 8.8M million in SegCNN). Fig. 3 shows the confusion matrix for our proposed model (with cycle graph).

Network size. Table 2 compares the number of learnable network parameters for various models with ours. All graph networks are smaller (an order of magnitude smaller) than LSTM architectures yet highly accurate. Our model has the highest accuracy with half the parameters of other graph-based networks. This is owing to the light-weight convolution operation and because of the choice of our graph structure. In our approach graph structure remains the same for all samples, which requires us to compute the eigen-decomposition only once. This operation can even be replaced by directly using DFT or DCT kernels.

Discussion. We obtained the best result using the cyclic graph structure. With the line graph, the model accuracy is slightly lower. We also performed experiments to investigate the contribution of the various components of our network. Table 3 compares performances for different pooling strategies used to compute graph-level representation from the node embeddings. As noted in a past work [21], sumpool improves results over meanpool and maxpool by 2.84% and 3.61% respectively. When using the convolution operation without MLP (see Eq. 4), performance drops by 1% (see Table 1). These results confirm that each component in our network contributes positively towards its performance.

4. Conclusion

We proposed a compact and efficient GCN-based architecture for recognizing emotion content in speech. To the best of our knowledge, this is the first graph-based approach to SER. We transform speech utterances to graphs with simple structure that largely simplifies the convolution operation in graph domain. Also, the graph structure we defined remains the same for all samples as our edges are not weighted. This leads to a light-weight GCN architecture which outperforms the traditional GCN [11] and other recent graph-based approaches [22, 26]. The proposed compact architecture produces state-of-the-art performance on the benchmark IEMOCAP database. Future work will be directed towards exploring graph weights learning within the network.
5. References


