Hybrid Message Passing with Performance-driven Structures for Facial Action Unit Detection Supplementary Material

Tengfei Song\textsuperscript{1}, Zijun Cui\textsuperscript{2}, Wenming Zheng\textsuperscript{1}, and Qiang Ji\textsuperscript{2}
\textsuperscript{1}Southeast University
\textsuperscript{2}Rensselaer Polytechnic Institute
songtf@seu.edu.cn, cuiz3@rpi.edu, wenming.zheng@seu.edu.cn, qji@ecse.rpi.edu

Figure 7. The framework to generate the AU features by using ResNet18.

1. The Generation of AU features

In this part, we introduce the details about how to extract disentangled AU features, which are further fed into message passing neural network. As shown in Figure 7, we use the ResNet18 to extract the disentangled AU features. \(X = x_1, \ldots, x_N \in \mathbb{R}^{d \times N}\) are \(N\) features corresponding for \(N\) AUs. In this paper, \(d\) is set to 512. \([x_1, \ldots, x_N]\) are reshaped by the output of ResNet18. Each feature vector is fed into a fully connected layer to predict the probability of the corresponding AU. The predicted probability for the \(i\)th AU can be calculated by

\[ p_i = \sigma(w_i^T x_i + b_i), \]

in which \(\sigma\) is the sigmoid function. \(w_i\) and \(b_i\) are the parameter of the \(i\)th full connection layer. To alleviate the influence of the imbalanced data, we calculated the weighted loss function to train the model and the loss function can be written as

\[ \mathcal{L} = - \sum_{i=1}^{N} \tau_i [y_i \log p_i + (1 - y_i) \log (1 - p_i)], \]

in which \(y_i\) denote the ground truth and \(\tau_i\) are constant weight. We directly use \(\tau_i\) provided in [1].

2. Pseudo Code to Sample Multiple Graph Structures

\begin{algorithm}
\caption{Obtain multiple graph structures \(\{A^n\}_{n=1}^N\) by Performance-based MCMC sampling}
\begin{algorithmic}
\Require Training data \(D\)
\Ensure \(N\) samples \(\{A^n\}_{n=1}^N\)
\State \textbf{Initial} graph structure \(A^1\);
\For {\(n = 1\) to \(N\)}
\State Generate a new sample \(A^{n+1}\) based on proposal probability Equ (1);
\State Fully train the neural network and estimate the performance, \(\text{i.e.,} \ \text{acc}(A^{n+1}, w)\).
\State Calculate the acceptance probability \(p_{\text{acc}}(A^{n+1})\) by Equ (4);
\If {accept}
\State \(A^{n+1} = A^{n+1}\);
\Else
\State \(A^{n+1} = A^n\);
\EndIf
\EndFor
\end{algorithmic}
\end{algorithm}

References