Compact Graph Architecture for Speech Emotion Recognition

Hamza Masood (210403)

Implementation Details

I have added an attention layer at the end and also explored the possibility of using kernel activation functions to have more flexibility. The implementation details of these modules are as follows:

Attention mechanism

The `SelfAttentionLayer` class computes relevance scores between features by learning relationships across nodes. It applies a learned linear transformation (W) to the feature matrix and then measures the importance of each node's features relative to others. The attention scores are computed through a leaky ReLU-activated comparison of transformed features, followed by normalization with softmax. This aggregated output then serves as a representation of the attended features. The attention mechanism is capturing essential relationships in the graph structure.

Kernel Activation Function

To enhance nonlinearity, the kernel activation functions (`KAF_random` and `KAF`) use Gaussian kernels to transform input data. These functions rely on a set of fixed kernel centers, spread around zero, which serve as dictionary points for Gaussian functions. The output of these functions is a weighted combination of Gaussian kernels, with weights (mixing coefficients) that are learned.

`KAF_random` initializes these weights (`alpha`) randomly, whereas `KAF` initializes `alpha` to approximate an Exponential Linear Unit (ELU) activation.

Results

$$\text{No. of trainable parameters} = \begin{cases} 38916, \text{without attention layer} \\ 43086, \text{with attention layer} \end{cases}$$

Graph Type	**Without Attention Layer**		**With Attention Layer**	
	WA (%)	**UA (%)**	**WA (%)**	**UA (%)**
Line	56.44	62.84	56.05	62.50
Cycle	56.19	62.05	55.65	62.05

Table 1: Results Comparison for Line and Cycle Graphs with and without Attention Layer (IEMOCAP)

Graph	**ReLU**	**Random KAF**	**KAF**
Line	62.84	62.00	60.66
Cycle	62.05	63.13	60.00

Table 2: Unweighted accuracy comparison for different activation functions

Pooling	Maxpool	Meanpool	Sumpool
WA (%)	55.03	58.53	56.49
UA (%)	62.16	64.72	63.17

Table 3: Comparing different pooling strategies on the IEMOCAP dataset

Activation	Without Attention Layer		With Attention Layer	
	WA (%)	**UA (%)**	**WA (%)**	**UA (%)**
KAF	54.29	60.66	51.83	58.70
ReLU	56.44	62.84	56.05	62.50
Random KAF	55.47	62.00	54.81	61.61

Table 4: Results comparison for different activation functions for Line graph with and without activation

Activation	Without Attention Layer		With Attention Layer	
	WA (%)	**UA (%)**	**WA (%)**	**UA (%)**
KAF	52.19	58.33	52.79	60.00
ReLU	56.19	62.05	55.65	62.05
Random KAF	56.76	63.13	54.53	61.48

Table 5: Results comparison for different activation functions for Cycle graph with and without activation

Inferences

The results indicate that adding an attention layer increases the model's trainable parameters (from 38,916 to 43,086) but does not improve the accuracy, suggesting that the attention mechanism may not be providing significant benefit for this particular graph-based structure on the IEMOCAP dataset, potentially indicating that the inherent structure captures relevant features well without needing additional focus through attention.

In terms of activation functions, the cycle graph with random KAF achieves the highest unweighted accuracy (63.13%), but for line graph ReLU performs better. Trying to mimic ELU doesn't seem to work for this case and performs worse than random initialization of the mixing coefficients.