1. Details on GNN Propagation in GgPC

Graph neural networks (GNN) are well established for the application [2, 3, 4, 1] of few-shot image classification. In our method, we followed EGGN [3] to utilize GNN as guidance to optimize the intra- and inter-class correlation within features. For simplicity and convenience, we discuss the process of the $N_S$-way 1-shot problem and consider that the query set $Q$ contains $N_Q$ videos. We let $G = (V, A; S \cup Q)$ be the graph to construct the relationship between support set videos $S$ and query videos $Q$. We use the video features as node features $V = \{v_i\}_{i=1, \ldots, |S \cup Q|}$ and the relationship between the node features as edge features $A = \{a_{ij}\}_{i,j=1, \ldots, |S \cup Q|}$, where $|S \cup Q| = N_S + N_Q$.

Node features are initialized by the enhanced temporal features after the mean pooling operation on the temporal dimension, i.e., $v_i^0 = F^{temp}(v_i \in S \cup Q)$. Edge features $a_{ij} \in \mathbb{R}^2(\forall i, j \in S \cup Q)$ are 2D vectors representing the intra- and inter-class relations of the two connected nodes and are initialized with ground-truth $y$, as follows:

$$a_{ij}^0 = \begin{cases} [1|0], & \text{if } y_i = y_j \text{ and } i, j \leq N_S, \\ [0|1], & \text{if } y_i \neq y_j \text{ and } i, j \leq N_S, \\ [0.5|0.5], & \text{otherwise}, \end{cases} \quad (1)$$

The $G$ consists of $L$ layers, and its propagation includes node features and edge features updating. Given $v_i^{l-1} \in \mathbb{R}^C$ and $a_{ij}^{l-1} \in \mathbb{R}^2$ from the layer $l - 1$, node features’ updating is a weighted aggregation process of other nodes through the layers’ edge features, as follows:

$$v_i^{l} = f_{node}^{l}(\text{Cat}(\sum_{j} a_{ij}^{l-1} v_j^{l-1}, \sum_{j} a_{ij}^{l-1} v_j^{l-1}), \text{dim}=0)) \quad (2)$$

where $f_{node}$ is a MLP to transform feature and $a_{ij}^{l-1} = \frac{a_{ij}^{l-1}}{\sum_b a_{ib}^{l-1}}$ on $b \in \{1, 2\}$. After the update of node features, the edge feature is updated through the (dis)similarities between two connected features, and the sum of all edge features’ values is kept constant, given by:

$$\bar{a}_{ij}^{l} = \begin{cases} \frac{f_{edge}^{l}(|v_i^l - v_j^l|) a_{ij}^{l-1}}{\sum_h f(|v_i^l - v_h^l|) a_{ih}^{l-1}} \sum_h a_{ih}^{l-1}, & \text{if } b = 0 \\ \frac{(1-f_{edge}^{l}(|v_i^l - v_j^l|)) a_{ij}^{l-1}}{\sum_h (1-f_{edge}^{l}(|v_i^l - v_h^l|)) a_{ih}^{l-1}} \sum_h a_{ih}^{l-1}, & \text{if } b = 1 \end{cases} \quad (3)$$

$$a_{ij}^{l} = \bar{a}_{ij}^{l}/\|\bar{a}_{ij}^{l}\|_1 \quad (4)$$

where $f_{edge}$ is a function to calculate the similarities between two connected nodes. Here we set $f_{edge}$ to a four-layer convolution block, where each layer comprises a $1 \times 1$ convolutional layer, batch normalization, and LeakyReLU activation function.

2. Implementation Details of Experimental Setup

2.1. Network Architectures

The kernel size for the 1D channel-wise temporal convolution in CTRM is set to 3. The settings of hyperparameters in each dataset are shown in Tab.1.

<table>
<thead>
<tr>
<th></th>
<th>Kinetics</th>
<th>SSv2</th>
<th>UCF101</th>
<th>HMDB51</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\gamma$</td>
<td>0.1</td>
<td>0.5</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>$\beta$</td>
<td>0.9</td>
<td>0.5</td>
<td>0.9</td>
<td>0.9</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>0.4</td>
<td>0.6</td>
<td>0.5</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Table 1. The settings of hyperparameters in each dataset.

2.2. Training and Inference

In HPM, when $T$ is set to 8, $L$ is calculated as 32. The total number of training steps is set to 10. Tab.2 presents the learning rate and other settings for various datasets. In this table, $lr$ refers to the learning rate, $st_{iter}$ indicates the number of iterations per step, $steps$ represents the number...
Figure 1. Attention visualization of our GgHM on UCF101 in the 5-way 1-shot setting. Corresponding to the original RGB images (left), the attention maps without LDTM modules (middle) are compared to the attention maps with our LDTM modules (right).

Table 2. The settings of hyperparameters in each dataset.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>$lr$</th>
<th>$st_{iter}$</th>
<th>steps</th>
<th>$LRS$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kinetics</td>
<td>2.2e-5</td>
<td>1000</td>
<td>[0,6,9]</td>
<td>[1,0.5,0.01]</td>
</tr>
<tr>
<td>SSv2</td>
<td>1e-4</td>
<td>7500</td>
<td>[0,6,8,9]</td>
<td>[1,0.5,0.1,0.01]</td>
</tr>
<tr>
<td>HMDB51</td>
<td>1e-4</td>
<td>1000</td>
<td>[0,2,3,5]</td>
<td>[1,0.5,0.1,0.01]</td>
</tr>
<tr>
<td>UCF101</td>
<td>5e-05</td>
<td>1500</td>
<td>[0,2,3,5]</td>
<td>[1,0.5,0.1,0.01]</td>
</tr>
</tbody>
</table>

of steps to change the learning rate when using the multi-step scheduler, and $LRS$ denotes the multiplication factor for updating the learning rate at each changing step.
3. Attention Visualization of our GgHM

Fig. 1 shows the attention visualization of our GgHM on UCF101 in the 5-way 1-shot setting. Compared to the original RGB images on the left, the attention maps without LDTM modules (in the middle) are contrasted against the attention maps with our LDTM modules (on the right). Attention maps generated without the LDTM module contain numerous irrelevant or distracting focus areas. For example, the frames in “HorseRiding” show attention to the background and extraneous objects, diverting focus from the action. In contrast, attention maps generated using the LDTM module strongly correlate with the subject acting. Specifically, the frames in “Skiing” focus on the skier, and the frames in “TennisSwing” focus on the tennis player. These observations provide empirical evidence of the effectiveness of our LDTM module in enhancing spatiotemporal representation.

References