Summary of Paper : Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition

Jihwan Kim
Advanced Technology Team, Scalawox

August 11, 2022

1 Introduction

Human action can be recognized from multiple modalities, such as appearance, depth, optical flows, and body skeletons. Among these modalities, dynamic human skeletons usually convey significant information that is complementary to others. However, the modeling of dynamic skeletons has received relatively less attention than that of appearance and optical flows. In this work, we systematically study this modalities, with an aim to develop a principled and effective method to model dynamic skeletons and leverage them for action recognition.

The dynamic skeleton modality can be naturally represented by a time series of human joint locations, in the form of 2D or 3D coordinates. Earlier methods of using skeletons for action recognition simply employ the joint coordinatesat individual time steps to form feature vectors, and apply temporal analysis thereon. The capability of these methods is limited as they do not explicitly exploit the spatial relationships among the joints, which are crucial for understanding human actions.

To move beyond such limitations, we need a new method that can automatically capture the patterns embedded in the spatial configuration of the joints as well as their temporal dynamics. However, the skeletons are in the form of graphs instead of a 2D or 3D grids, which makes it difficult to use proven models like convolutional Networks.

In this paper, we propose to design a generic representation of skeleton sequences for action recognition byextending graph neural networks to a spatial-temporal graph works (ST-GCN).

There are two types of edges, namely the spatial edges that conform to the natural connectivity of joints and the temporal edges that connect the same joints across consecutive time steps.

Skeleton based data can be obtained from motion-capture devices or pose estimation algorithms from videos. Given the sequences of body joints in the form of 2D or 3D coordinates, we construct a spatial temporal graph with the joints as graph nodes and natural connectivities in both human body structures and time as graph edges. The input to the ST-GCN is therefore the joint coordinates vectors on the graph nodes. This can be considered as an analog to image based CNNs where the input is formed by pixel intensity vectors residing on the 2D image grid.

2 Spatial Temporal Graph ConvNet

2.1 Skeleton Graph Construction

Previous work using convulution for skeleton action recognition (Kim and Reiter 2017) concatenates coordinate vectors of all joints to form a single feature vector per frame. In our work, we utilize the spatial temporal graph to form hierarchical representation of the skeleton sequences. Particularly, we construct an undirected spatial temporal graph $G = (V, E)$ on a skeleton sequence with N joints and T frames featuring both intra-body and inter-frame connection.

In this graph, the node set $V = \{v_{ti} : t = 1, ..., T, i = 1, ..., N\}$ includes the all the joints in a skeleton sequence. As ST-GCN's input, the feature vector on a node $F(v_{ti})$ consists of coordinate vectors, as well as estimation confidence, of the ith joint on frame t.

Formally, the edge set E is composed of two subsets, the first subset depicts the intra-skeleton connection at each frame, denotes as $E_S = \{v_{ti}v_{tj} : (i,j) \in H\}$, where H is the set of naturally connected human body joints. The second subset contains the inter-frame edges, which connect the same joints in consecutive frames as $E_F = \{v_{ti}v_{(t+1)i}\}$.

2.2 Spatial Graph Convolutional Neural Network

Before we dive into the full fledged ST-GCN, we first look at the graph CNN model within one single frame. In this case, on a single frame at time τ, there will be N joint nodes V_t, along with the skeleton edges $E_S(\tau) = \{v_{ti}v_{tj} : t = \tau, (i,j) \in H\}$.

Recall the definition of convolution operation on the 2D natural images, or feature maps, which can be both treated as 2D grids. The output feature map of a convolution operation is again a 2D grid. Given a convolution operator with the kernel size of $K \times K$, and an input feature map f_{in} with the number of channels c. The output value for a single channel at the spatial location x can be written as

$$f_{out}(\mathbf{x}) = \sum_{h=1}^{K}\sum_{w=1}^{K} f_{in}(\mathbf{p}(\mathbf{x},h,w)) \cdot \mathbf{w}(h,w) \tag{1}$$

where the sampling function $\mathbf{p} : Z^2 \times Z^2 \to Z^2$ enumerates the neighbors of location $\mathbf{x}$. In the case of image convolution, it can also be represented as $\mathbf{p}(\mathbf{x},h,w) = \mathbf{x} + \mathbf{p}'(h,w)$. The weight function $\mathbf{w} : Z^2 \to \mathbb{R}^c$ provides a weight vector in c-dimension real space for computing the inner product with the sampled input feature vectors of dimension c. Standard convoultion on the image domain is therefore achieved by encoding a rectangular grid in $\mathbf{p}(\mathbf{x})$.

The convulution operation on graphs is then defined by extending the formulation above to the cases where the input features map resides on a spatial graph V_t. That is, the feature map $f_{in}^t : V_t \to \mathbb{R}^c$ has a vector on each node of the graph.

Sampling function On graphs, we can similarly define the sampling function on the neighbor set $B(v_{ti}) = \{v_{tj} : d(v_{tj}, v_{ti}) \leq D\}$ of a node v_{ti}. Thus the sampling function $\mathbf{p} : B(v_{ti}) \to V$ can be written as

$$\mathbf{p}(v_{ti}, v_{tj}) = v_{tj}. \tag{2}$$

In this work we use $D = 1$ for all cases, that is, the 1-neighbor set of joint nodes.

Weight function In 2D convolution, a rigid grid naturally exists around the center location. So pixeles within the neighbor can have a fixed spatial order. The wiehgt function can then be implemented by indexing a tensor of (c, K, K) dimensions according to the spatial order. For general graphs, there is no such implicit arrangement. The solution to this problem is first investigated in (Niepert, Ahmed, and Kutzkov 2016), where the order is defined by a graph labeling process in the neighbor graph around the root node. We follow this idea to construct our weight function. Instead of giving every neighbor node a unique labeling, we simplify the process by partitioning the neighbor set $B(v_{ti})$ of a joint node v_{ti} into a fixed number of K subsets, where each subset has a numeric label. Thus we can have a mapping $l_{ti} : B(v_{ti}) \to \{0, ..., K-1\}$ which maps a node in the neighborhood to its subset label. The weight function $\mathbf{w}(v_{ti}, v_{tj}) : B(v_{ti}) \to \mathbb{R}^c$ can be implemented by indexing a tensor of (c, K) dimension or

$$\mathbf{w}(v_{ti}, v_{tj}) = \mathbf{w}'(l_{ti}(v_tj)). \tag{3}$$

Spatial Graph Convulution With the refined sampling function and weight function, we now rewrite Eq. (1) in terms of graph convulution as

$$f_{out}(v_{ti}) = \sum_{v_{tj} \in B(v_{ti})} \frac{1}{Z_{ti}(v_{tj})} f_{in}(\mathbf{p}(v_{ti}, v_{tj})) \cdot \mathbf{w}(v_{ti}, v_{tj}). \tag{4}$$

where the normalizing term $Z_{ti}(v_{tj}) = |\{v_{tk} : l_{ti}(v_{tk}) = l_{ti}(v_{tj})\}|$ equals the cardinality of the corresponding subset. Substituting Eq. 2 and Eq. 3 into Eq. 4, we arrive at

$$f_{out}(v_{ti}) = \sum_{v_{tj} \in B(v_{ti})} \frac{1}{Z_{ti}(v_{tj})} f_{in}(v_{tj}) \cdot \mathbf{w}(l_{ti}(v_{tj})). \tag{5}$$

Spatial Temporal Modeling Having formulated spatial graph CNN, we now advance to the task of modeling the spatial temporal dynamics within skeleton sequence. We extend the concept of neighborhood to also include temporally connected joints as

$$B(v_{ti}) = \{v_{qj} : d(v_{qj}, v_{ti}) \leq K, |q - t| \leq \lfloor \Gamma/2 \rfloor\}. \tag{6}$$

The parameter Γ controls the temporal range to be included in the neighbor graph and can thus be called the temporal kernel size. To complete the convolution operation on the spatial temporal graph, we also need the sampling function, which is the same as the spatial only case, and the weight function, or in particular, the labeling map l_{ST}. Because the temporal axis is well-ordered, we directly modify the label map l_{ST} for a spatial temporal neighborhood rooted at v_{ti} to be

$$l_{ST}(v_{qj}) = l_{ti}(v_{tj}) + (q - t + \lfloor \Gamma/2 \rfloor) \times K. \tag{7}$$

where $l_{ti}(v_{tj})$ is the label map for the single frame case at v_{ti}.

2.3 Partition Strategies

Uni-labeling The simplest and most straight forward partition strategy is to have subset, which is the whole neighbor set itself. In this strategy, feature vectors on every neighboring node will have a inner product with the same weight vector. This is suboptimal for skeleton sequence classification as the local differential properties could be lost in this operation. Formally, we have $K = 1$ and $l_{ti}(v_{tj}) = 0, \forall i, j \in V$

Distance partitioning Another natural partitioning strategy is to partition the neighbor set according to the nodes' distance $d(\cdot, v_{ti})$ to the root node v_{ti}. In this work, because we set $D = 1$, the neighbor set will then be separaed into two subsets, where $d = 0$ refers to the root node itself and remaining neighbor nodes are in the $d = 1$ subset. Formally, we have $K = 2$ and $l_{ti}(v_{tj}) = d(v_{tj}, v_{ti})$.

Spatial configuration patitioning Since the body skeleton is spatially localized, we can still utilize this specific spatial configuration in the partitioning process. We design a strategy to divide the neighbor set into three subsets: 1) the root node itself; 2)centripetal group: the neighboring nodes that are closer to the gravity center of the skeleton than the root nodes; 3) otherwise the centrifugal group. Here the average coordinate of all joints in the skeleton at a frame is treated as its gravity center. Formally, we have

$$l_{ti}(v_{tj}) = \begin{cases} 0 & \text{if } r_j = r_i \\ 1 & \text{if } r_j < r_i \\ 2 & \text{otherwise} \end{cases} \tag{8}$$

where r_i is the average distance from gravity center to joint i over all frames in the training set.

2.4 Learnable edge importance weighting

Although joints move in groups when people are performing actions, one joint could appear in multiple body parts. These appearances, however, should have different importance in modeling the dynamics of these parts. In this sennse, we add a learnable mask M on every layer of spatial temporal graph convolution. The mask will scale the contribution of a node's feature to its neighboring nodes based on the learned importance weight of each spatial graph edge in E_S.

2.5 Implementing ST-GCN

We adopt a similar implementation of graph convolution as in (Kipf and Welling 2017). The intra body connections of joints within a single frame are represented by an adjacency matrix $\mathbf{A}$ and an identity matrix I representing self-connections. In the single frame cae, ST-GCN with the first partitioning strategy can be implemented with the following formula

$$f_{out} = \Lambda^{-1/2}(\mathbf{A} + \mathbf{I})\Lambda^{-1/2} f_{in}\mathbf{W}. \tag{9}$$

where $\Lambda^{ii} = \sum_j (A^{ij} + I^{ij})$. In practice, under the spatial temporal cases, we can represent the input feature map as a tensor of (C, V, T) dimensions. The graph convolution is implemented by performing a $1 \times \Gamma$ standard 2D convolution and multiplies the resulting tensor with the normalized adjacency matrix $\mathbf{\Lambda}^{-1/2}(\mathbf{A} + \mathbf{I})\mathbf{\Lambda}^{-1/2}$ on the second dimension.

For partitioning strategies with multiple subsets, we again utilize this implementation. But note now the adjacency matrix is dismantled into several matrixes $\mathbf{A}_j$ where $\mathbf{A} + \mathbf{I} = \sum_j \mathbf{A}_j$. For example in the distance partitioning strategy, $\mathbf{A}_0 = \mathbf{I}$ and $\mathbf{A}_1 = \mathbf{A}$. Eq. 9 is transformed into

$$f_{out} = \sum_j \mathbf{\Lambda}_j^{-1/2}\mathbf{A}_j\mathbf{\Lambda}_j^{-1/2} f_{in}\mathbf{W}_j, \tag{10}$$

where similarly $\Lambda_j^{ii} = \sum_k (A_j)^{ik} + \alpha$. Here we set $\alpha = 0.001$ to avoid empty rows in $\mathbf{A}_j$.

It is straightforward to implement the learnable edge importance weighting. For each adjacency matrix, we accompany it with a learnable weight matrix $\mathbf{M}$. And we substitute the matrix $\mathbf{A} + \mathbf{I}$ in Eq. 9 and A_j in $\mathbf{A}_j$ in Eq. 10 with $(\mathbf{A} + \mathbf{I}) \otimes \mathbf{M}$ and $\mathbf{A}_j \otimes \mathbf{M}$, respectively. Here $\otimes$ denotes element-wise product between two matrixes. The mask $\mathbf{M}$ is initialized as an all-one matrix.

Network architecture and training Since the ST-GCN share weights on different nodes, it is important to keep the scale of input date consistent on different joints. In our experiments, we first feed input skeletons to a batch normalization layer to normalize data. The ST-GCN model is composed of 9 layers of spatial temporal graph convolution operators. The first three layers have 64 channels for output. The follow three layers have 128 channels for output. And the last three layers have 256 channels for output. These layers have 9 temporal kernel size. The Resnet mechanism is applied on each ST-GCN unit. And we randomly dropout the features at 0.5 probability after each ST-GCN unit to avoid overfitting. The strides of the 4th and the 7th temporal convolution layers are set to 2 as pooling layer. After that, a global pooling was performed on the resulting tensor to get a 256 dimension feature vector for each sequence. After that a global pooling was performed on the resulting tensor to get a 256 dimension feature vector for each sequence. Finally, we feed them to a SoftMax classifier. The models are learned using stochastic gradient descent with a learning rate of 0.01. We decaay the learning rate by 0.1 after every 10 epochs.