UCLA

Progress of Graph Convolutional Networks

[Cheng Zheng]
[04/27/2018]

UCLA

Roadmap

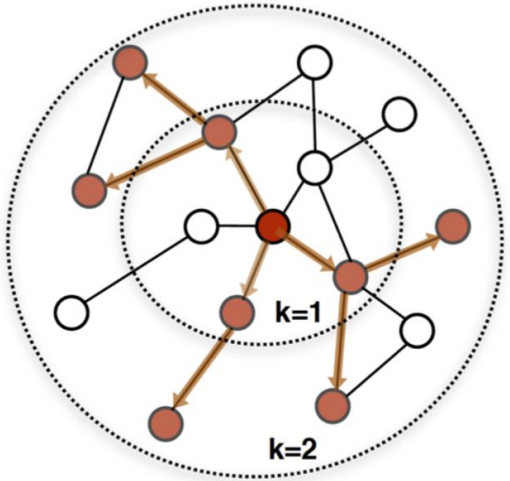
- GCN (NIPS 16', ICLR 17')
- FastGCN (ICLR 18')
- ST-GCN (AAAI 18')

UCLA

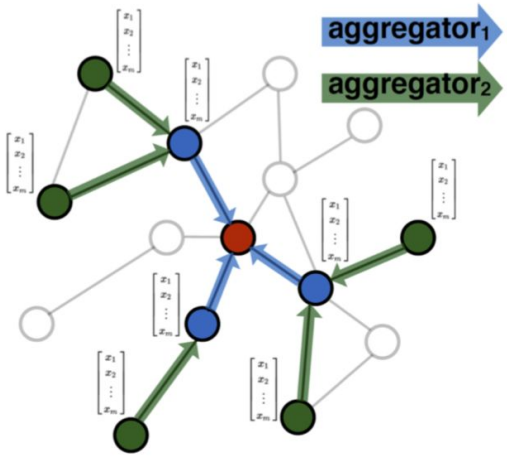
Roadmap

- GCN (NIPS 16', ICLR 17')
- FastGCN (ICLR 18')
- ST-GCN (AAAI 18')

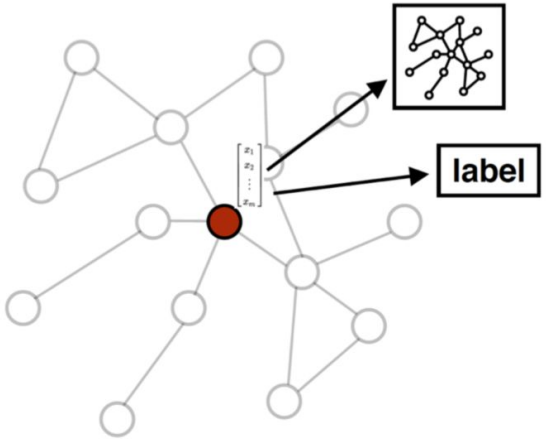
Intuitive GCN



1. Sample neighborhood



2. Aggregate feature information from neighbors



3. Predict graph context and label using aggregated information

Hamilton, W., Ying, Z., & Leskovec, J. Inductive representation learning on large graphs. NIPS 2017

UCLA

GCN Approximations

If K=2, filtering becomes: $g_{\theta'} \star x \approx \theta'_0 x + \theta'_1 \left(L - I_N\right) x = \theta'_0 x - \theta'_1 D^{-\frac{1}{2}} A D^{-\frac{1}{2}} x$

If set (further approximate): $\theta = \theta'_0 = -\theta'_1.$

Then, filtering becomes: $g_\theta \star x \approx \theta \left(I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}\right) x$

If input is a matrix: $X \in \mathbb{R}^{N \times C}$

Then, filtering becomes: $Z = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} X \Theta$

Filter parameters: $\Theta \in \mathbb{R}^{C \times F}$

Convolved signal matrix: $Z \in \mathbb{R}^{N \times F}$

Filtering complexity: $\mathcal{O}(|\mathcal{E}|FC)$

Slides borrowed from Yunsheng

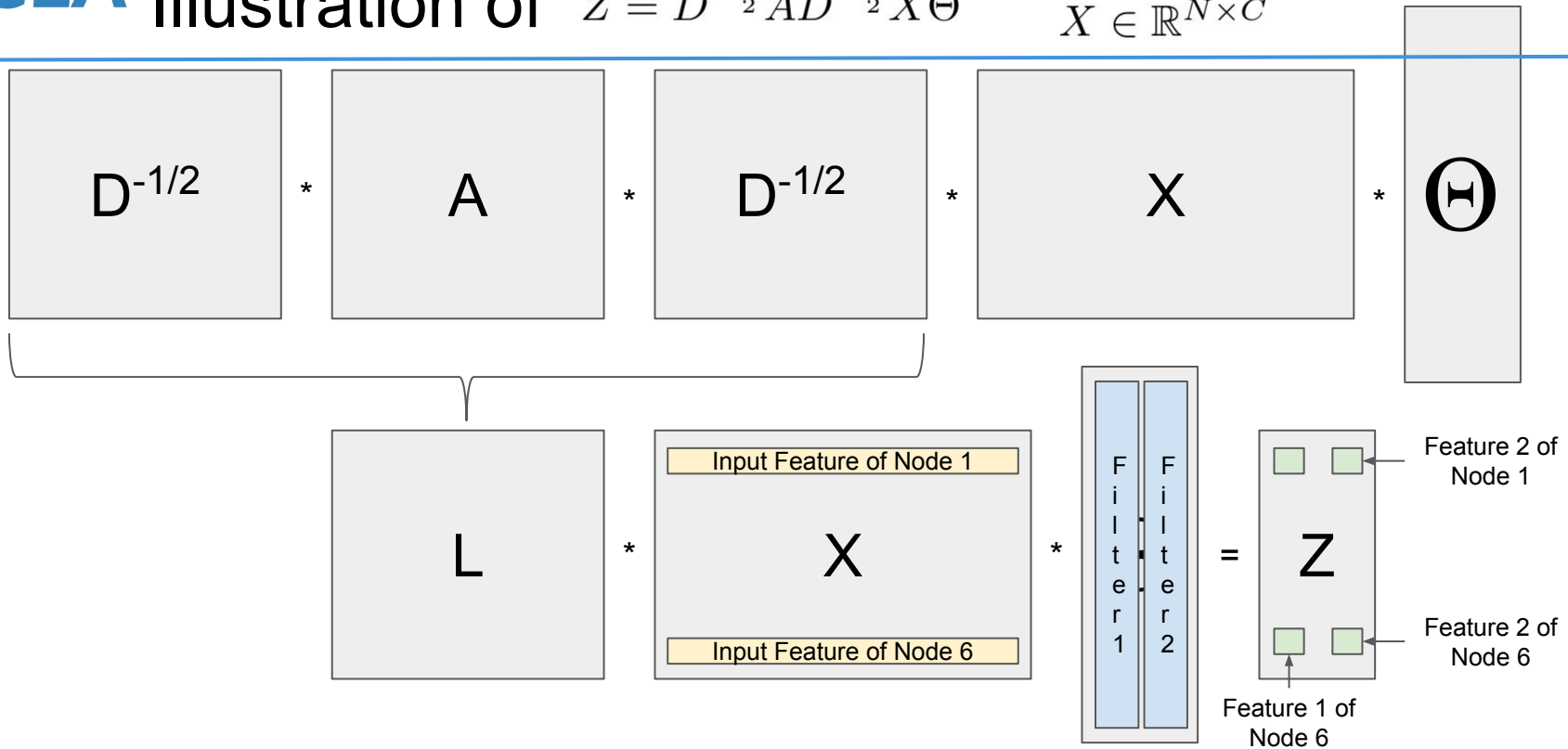
Semi-Supervised Classification with Graph Convolutional Networks (ICLR 2017)

UCLA

Illustration of $Z = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}X\Theta$

$\Theta \in \mathbb{R}^{C\times F}$

$X \in \mathbb{R}^{N\times C}$

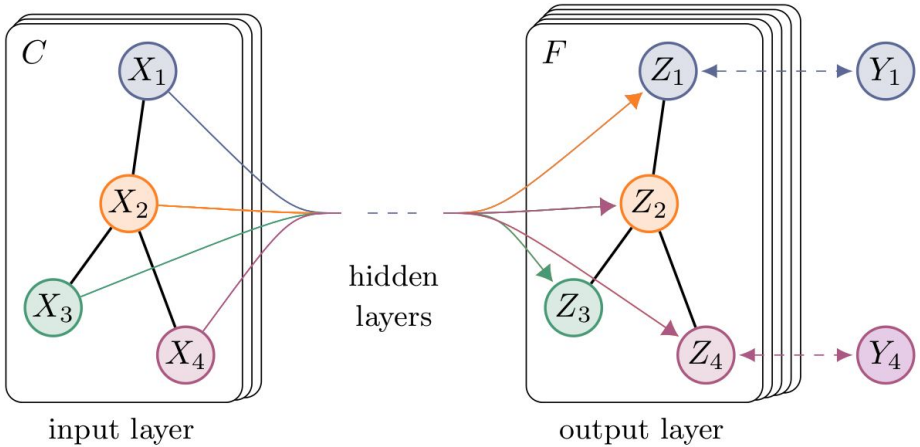


$Z \in \mathbb{R}^{N\times F}$

Semi-Supervised Classification with Graph Convolutional Networks (ICLR 2017)

Slides borrowed from Yunsheng

GCN Schematic Depiction



$$Z = f(X, A) = \mathrm{softmax}\Big(\hat{A}\ \mathrm{ReLU}\Big(\hat{A}XW^{(0)}\Big)\,W^{(1)}\Big)$$

$$\mathcal{L} = -\sum_{l \in \mathcal{Y}_L} \sum_{f=1}^{F} Y_{lf} \ln Z_{lf}$$

Slides borrowed from Yunsheng

Semi-Supervised Classification with Graph Convolutional Networks (ICLR 2017)

UCLA

Roadmap

- GCN (NIPS 16', ICLR 17')
- FastGCN (ICLR 18')
- ST-GCN (AAAI 18')

Chen, Jie, Tengfei Ma, and Cao Xiao. "FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling." ICLR, 2018.

UCLA

FastGCN

Motivations:

- Relax the requirement of simultaneous availability of test data.
- Time and memory challenge for training large, dense graphs.

Typical GCN layer operation,

$$H^{(l+1)} = \sigma(\hat{A}H^{(l)}W^{(l)})$$

Require an inductive scheme that learns a model from only a training set of vertices and generalizes well to expanding graph.

Chen, Jie, Tengfei Ma, and Cao Xiao. "FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling." ICLR, 2018.

UCLA

Lack of Independence

Standard Stochastic gradient descent(SGD):

Loss is the expectation of some function g w.r.t. a data distribution D

$$L = \mathrm{E}_{x \sim D}[g(W; x)].$$

With Unknown D,instead minimizes the empirical loss through accessing n iid samples x1,...,xn

$$L_{\text{emp}} = \frac{1}{n} \sum_{i=1}^{n} g(W; x_i), \qquad x_i \sim D, \ \forall i.$$

However, not available in graphs: node information are recursively linked together with their neighbors.

UCLA

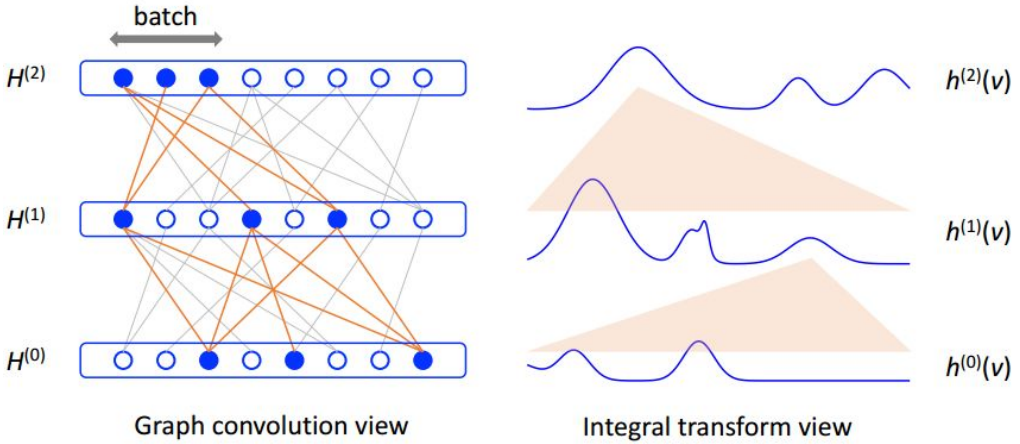
Alternative Formulation

A (possibly infinite) graph G’

- vertex set V’
- probability space (V’, F, P),

For the probability space, V’ serves as the sample space and F may be any event space (e.g., the power set $F = 2^{V'}$). The probability measure P defines a sampling distribution.

Integral Transform



Graph Convolution view

$$\tilde{H}^{(l+1)} = \hat{A}H^{(l)}W^{(l)}, \quad H^{(l+1)} = \sigma(\tilde{H}^{(l+1)}), \quad l = 0, \ldots, M-1, \quad L = \frac{1}{n}\sum_{i=1}^{n} g(H^{(M)}(i,:)). \tag{1}$$

For the functional generalization, we write

Integral Transform View

$$\tilde{h}^{(l+1)}(v) = \int \hat{A}(v,u)h^{(l)}(u)W^{(l)}\, dP(u), \quad h^{(l+1)}(v) = \sigma(\tilde{h}^{(l+1)}(v)), \quad l = 0, \ldots, M-1, \tag{2}$$

$$L = \mathrm{E}_{v\sim P}[g(h^{(M)}(v))] = \int g(h^{(M)}(v))\, dP(v). \tag{3}$$

UCLA

Monte Carlo Manner

For each layer l, t_l iid samples $u_1^{(l)},\ldots,u_{t_l}^{(l)} \sim P$ are used to approximately evaluate the integral transform

$$\tilde{h}^{(l+1)}_{t_{l+1}}(v) := \frac{1}{t_l}\sum_{j=1}^{t_l} \hat{A}(v,u_j^{(l)})h^{(l)}_{t_l}(u_j^{(l)})W^{(l)}, \quad h^{(l+1)}_{t_{l+1}}(v) := \sigma(\tilde{h}^{(l+1)}_{t_{l+1}}(v)), \quad l = 0,\ldots,M-1,$$

Then the loss is:

$$L_{t_0,t_1,\ldots,t_M} := \frac{1}{t_M}\sum_{i=1}^{t_M} g(h^{(M)}_{t_M}(u_i^{(M)})).$$

Theorem 1: If g and σ are continuous, then

$$\lim_{t_0,t_1,\ldots,t_M\to\infty} L_{t_0,t_1,\ldots,t_M} = L \quad \textit{with probability one.}$$

UCLA

Batched GCN

Uniformly sampling:

For each batch, we sample (with replacement) uniformly each layer and obtain samples $u_i^{(l)}$, $i = 1,\ldots,t_l$, $l = 0,\ldots,M-1$.

$$L_{\text{batch}} = \frac{1}{t_M} \sum_{i=1}^{t_M} g(H^{(M)}(u_i^{(M)}, :)),$$

$$H^{(l+1)}(v, :) = \sigma\left(\frac{n}{t_l} \sum_{j=1}^{t_l} \hat{A}(v, u_j^{(l)}) H^{(l)}(u_j^{(l)}, :) W^{(l)} \right), \quad l = 0, \ldots, M-1.$$

The corresponding batch gradient may be straightforwardly obtained through applying the chain rule on each $H^{(l)}$

UCLA

Important Sampling

Specifically, consider for the lth layer, the function $\tilde{h}_{t_{l+1}}^{(l+1)}(v)$ as an approximation to the convolution $\int \hat{A}(v,u)h_{t_l}^{(l)}(u)W^{(l)}\,dP(u)$. When taking t_{l+1} samples $v = u_1^{(l+1)}, \ldots, u_{t_{l+1}}^{(l+1)}$, the sample average of $\tilde{h}_{t_{l+1}}^{(l+1)}(v)$ admits a variance that captures the deviation from the eventual loss contributed

Computing the full variance is highly challenging because of nonlinearity in all the layers, consider to improve the variance of the embedding function before nonlinearity.

UCLA

Important Sampling cont.

	Function	Samples	Num. samples
Layer $l+1$; random variable v	$\tilde{h}^{(l+1)}_{t_{l+1}}(v) \to y(v)$	$u_i^{(l+1)} \to v_i$	$t_{l+1} \to s$
Layer l; random variable u	$h^{(l)}_{t_l}(u)W^{(l)} \to x(u)$	$u_j^{(l)} \to u_j$	$t_l \to t$

$$G := \frac{1}{s}\sum_{i=1}^{s} y(v_i) = \frac{1}{s}\sum_{i=1}^{s}\left(\frac{1}{t}\sum_{j=1}^{t}\hat{A}(v_i,u_j)x(u_j)\right)$$

Proposition 2. *The variance of G admits*

$$\mathrm{Var}\{G\} = R + \frac{1}{st}\iint \hat{A}(v,u)^2 x(u)^2\, dP(u)\, dP(v), \tag{6}$$

where

$$R = \frac{1}{s}\left(1-\frac{1}{t}\right)\int e(v)^2\, dP(v) - \frac{1}{s}\left(\int e(v)\, dP(v)\right)^2 \quad \textit{and} \quad e(v) = \int \hat{A}(v,u)x(u)\, dP(u).$$

UCLA

Important Sampling cont.

Important Sampling: sample t vertices u1, . . . ,ut according to this distribution.

$$q(u) = \|\hat{A}(:,u)\|^2 / \sum_{u' \in V} \|\hat{A}(:,u')\|^2, \quad u \in V$$

From the expression of q, we see that it has no dependency on l; that is, the sampling distribution is the same for all layers.

Algorithm 2 FastGCN batched training (one epoch), improved version

1: For each vertex u, compute sampling probability $q(u) \propto \|\hat{A}(:,u)\|^2$
2: **for** each batch **do**
3: For each layer l, sample t_l vertices $u_1^{(l)}, \ldots, u_{t_l}^{(l)}$ according to distribution q
4: **for** each layer l **do** ▷ Compute batch gradient ∇L_{batch}
5: If v is sampled in the next layer,

$$\nabla \tilde{H}^{(l+1)}(v,:) \leftarrow \frac{1}{t_l} \sum_{j=1}^{t_l} \frac{\hat{A}(v,u_j^{(l)})}{q(u_j^{(l)})} \nabla \left\{ H^{(l)}(u_j^{(l)},:) W^{(l)} \right\}$$

6: **end for**
7: $W \leftarrow W - \eta \nabla L_{\text{batch}}$ ▷ SGD step
8: **end for**

Inference can be done with normal GCN because of trained parameter and easy forward-propagation.

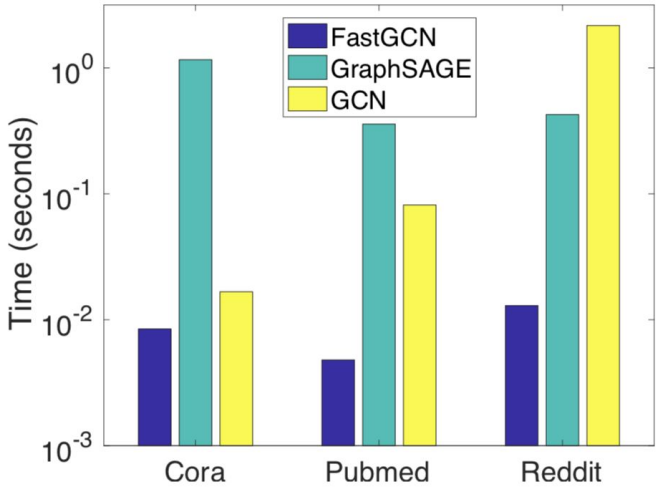
Experiment

Dataset	Nodes	Edges	Classes	Features	Training/Validation/Test
Cora	2,708	5,429	7	1,433	1,208/500/1,000
Pubmed	19,717	44,338	3	500	18,217/500/1,000
Reddit	232,965	11,606,919	41	602	152,410/23,699/55,334

Node Classification Task

Micro F1 Score

	Cora	Pubmed	Reddit
FastGCN	0.850	0.880	0.937
GraphSAGE-GCN	0.829	0.849	0.923
GraphSAGE-mean	0.822	0.888	0.946
GCN (batched)	0.851	0.867	0.930
GCN (original)	0.865	0.875	NA



UCLA

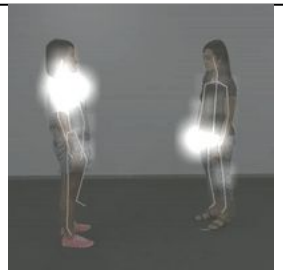
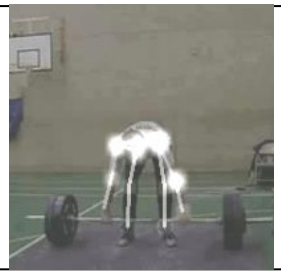
Roadmap

- GCN (NIPS 16', ICLR 17')
- FastGCN (ICLR 18')
- ST-GCN (AAAI 18')

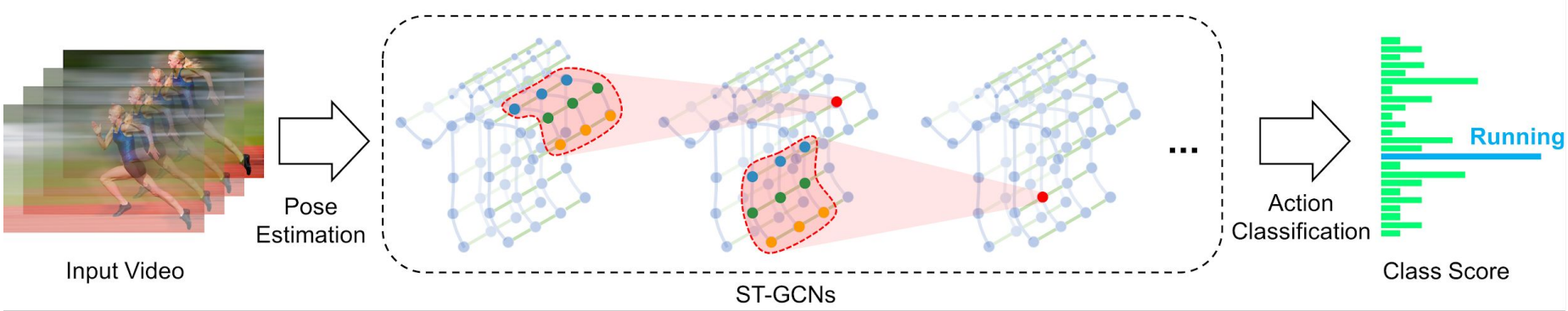
Yan, S., Xiong, Y., & Lin, D. Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. AAAI 2018

UCLA

Skeleton Sequences

				
Touch head	Sitting down	Take off a shoe	Eat meal/snack	Kick other person
				
Hammer throw	Clean and jerk	Pull ups	Tai chi	Juggling ball

Spatial Temporal Graph ConvNet



Yan, S., Xiong, Y., & Lin, D. Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. AAAI 2018

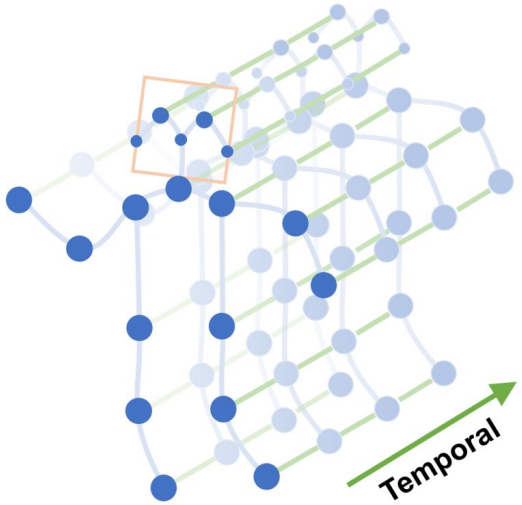
Skeleton Graph Construction

Skeleton data:

- from motion-capture devices or pose estimation algorithms on videos.
- sequence of frames, each frame will have a set of joint coordinates.

Undirected spatial temporal graph G = (V, E):

- N joints and T frames
- Node set $V = \{v_{ti} = 1,...,T, i = 1,...,N\}$
- Intra-skeleton connection $E_S = \{v_{ti}v_{tj} | (i, j) \in H\}$
- Each joint will be connected to the same joint in the consecutive frame $E_F = \{v_{ti}v_{(t+1)}i\}$.



Model

output value for a single channel at the node v_{ti}

Graph labeling process in the neighbor graph around the root node.
Next slide ->

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

Node sampling method:

- Nearest neighbor $B(v_{ti}) = \{v_{tj} | d(v_{tj}, v_{ti}) \leq D\}$, $D = 1$
- Spatial Temporal Modeling: $B(v_{ti}) = \{v_{qj} | d(v_{tj}; v_{ti}) \leq K; |q - t| \leq \Gamma/2\}$:

normalizing term

Input feature

Label Map Strategies

b) Uni-labeling: same weight for neighboring nodes

c) Distance partitioning: partition the neighbor set according to the nodes' distance to the root node

d) Spatial configuration partitioning: specific spatial configuration in the partitioning process

1. The root node 2. centripetal group (closer to the gravity center) 3. centrifugal group

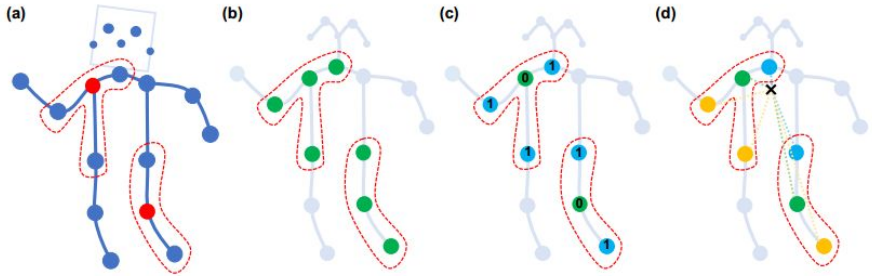


Figure 3: The proposed partitioning strategies for constructing convolution operations. From left to right: **(a)** An example frame of input skeleton. Body joints are drawn with blue dots. The receptive fields of a filter with $D = 1$ are drawn with red dashed circles. **(b) Uni-labeling** partitioning strategy, where all nodes in a neighborhood has the same label (green). **(c) Distance** partitioning. The two subsets are the root node itself with distance 0 (green) and other neighboring points with distance 1. (blue). **(d) Spatial configuration** partitioning. The nodes are labeled according to their distances to the skeleton gravity center (black cross) compared with that of the root node (green). Centripetal nodes have shorter distances (blue), while centrifugal nodes have longer distances (yellow) than the root node.

UCLA

Datasets

Kinetics human action dataset: 300; 000 raw video clips retrieved from YouTube. 400 human action classes, ranging from daily activities, sports scenes, to complex actions with interactions. Each clip in Kinetics lasts around 10 seconds. Preprocessed with *OpenPose* toolbox to get 18 joints on every frame of the clips in 2D coordinates.

NTURGB+D: 3D joints annotations for human action recognition task. This dataset contains 56,000 action clips in 60 action classes. 25 joints for each subject.

UCLA

Experiments

	Top-1	Top-5
RGB(Kay et al. 2017)	57.0%	77.3%
Optical Flow (Kay et al. 2017)	49.5%	71.9%
Feature Enc. (Fernando et al. 2015)	14.9%	25.8%
Deep LSTM (Shahroudy et al. 2016)	16.4%	35.3%
Temporal Conv. (Kim and Reiter 2017)	20.3%	40.0%
ST-GCN	**30.7%**	**52.8%**

Table 2: Action recognition performance for skeleton based models on the Kinetics dataset. On top of the table we list the performance of frame based methods.

	X-Sub	X-View
Lie Group (Veeriah, Zhuang, and Qi 2015)	50.1%	52.8%
H-RNN (Du, Wang, and Wang 2015)	59.1%	64.0%
Deep LSTM (Shahroudy et al. 2016)	60.7%	67.3%
PA-LSTM (Shahroudy et al. 2016)	62.9%	70.3%
ST-LSTM+TS (Liu et al. 2016)	69.2%	77.7%
Temporal Conv (Kim and Reiter 2017).	74.3%	83.1%
C-CNN + MTLN (Ke et al. 2017)	79.6%	84.8%
ST-GCN	**81.5%**	**88.3%**

Table 3: Skeleton based action recognition performance on NTU-RGB+D datasets. We report the accuracies on both the cross-subject (X-Sub) and cross-view (X-View) benchmarks.

	Top-1	Top-5
Baseline TCN	20.3%	40.0%
Local Convolution	22.0%	43.2%
Uni-labeling	19.3%	37.4%
Distance partitioning*	23.9%	44.9%
Distance Partitioning	29.1%	51.3%
Spatial Configuration	29.9%	52.2%
ST-GCN + Imp.	**30.7%**	**52.8%**

Table 1: Ablation study on the Kinetics dataset. The "ST-GCN+Imp." is used in comparison with other state-of-the-art methods. For meaning of each setting please refer to Sec 4.2.

Method	RGB CNN	Flow CNN	ST-GCN
Accuracy	70.4%	72.8%	72.4%

Table 4: Mean class accuracies on the "Kinetics Motion" subset of the Kinetics dataset. This subset contains 30 action classes in Kinetics which are strongly related to body motions.

UCLA

Thank you!

Q & A

UCLA