Learning Connectivity with Graph Convolutional Networks

Hichem SAHBI

CNRS Sorbonne University, Paris

ICPR 2020
Outline

- Introduction
- Learning connectivity in GCNs
- Experiments
- Conclusion
Outline

1 Introduction

2 Learning connectivity in GCNs

3 Experiments

4 Conclusion
Graph convolutional networks (GCNs) aim at generalizing deep learning to arbitrary irregular domains.

Existing spatial GCNs follow a neighborhood aggregation scheme, and its success is reliant on the topology (structure) of input graphs.

However, graph structures (either available or handcrafted) are powerless to optimally capture all the relationships between nodes as their setting is oblivious to the targeted applications.

E.g., node-to-node relationships, in human skeletons, capture the intrinsic anthropometric characteristics of individuals (useful for their identification) while other connections, yet to infer, are necessary for recognizing their dynamics and actions.
Graph convolutional networks (GCNs) aim at generalizing deep learning to arbitrary irregular domains.

Existing spatial GCNs follow a neighborhood aggregation scheme, and its success is reliant on the topology (structure) of input graphs.

However, graph structures (either available or handcrafted) are powerless to optimally capture all the relationships between nodes as their setting is oblivious to the targeted applications.

E.g., node-to-node relationships, in human skeletons, capture the intrinsic anthropometric characteristics of individuals (useful for their identification) while other connections, yet to infer, are necessary for recognizing their dynamics and actions.
Graph convolutional networks (GCNs) aim at generalizing deep learning to arbitrary irregular domains.

Existing spatial GCNs follow a neighborhood aggregation scheme, and its success is reliant on the topology (structure) of input graphs.

However, graph structures (either available or handcrafted) are powerless to optimally capture all the relationships between nodes as their setting is oblivious to the targeted applications.

E.g., node-to-node relationships, in human skeletons, capture the intrinsic anthropometric characteristics of individuals (useful for their identification) while other connections, yet to infer, are necessary for recognizing their dynamics and actions.
Graph convolutional networks (GCNs) aim at generalizing deep learning to arbitrary irregular domains.

Existing spatial GCNs follow a neighborhood aggregation scheme, and its success is reliant on the topology (structure) of input graphs.

However, graph structures (either available or handcrafted) are powerless to optimally capture all the relationships between nodes as their setting is oblivious to the targeted applications.

E.g., node-to-node relationships, in human skeletons, capture the intrinsic anthropometric characteristics of individuals (useful for their identification) while other connections, yet to infer, are necessary for recognizing their dynamics and actions.
We introduce a novel framework that learns convolutional filters on graphs together with their topological properties. The latter are modeled through matrix operators that capture multiple aggregates on graphs, learned using a constrained cross-entropy loss. We consider different constraints (including stochasticity, orthogonality and symmetry) acting as regularizers. Stochasticity implements random walk Laplacians while orthogonality models multiple aggregation operators with non-overlapping supports; it also avoids redundancy and oversizing the learned GCNs with useless parameters. Symmetry reduces further the number of training parameters.
Contribution: learning connectivity in GCNs

- We introduce a novel framework that learns convolutional filters on graphs together with their topological properties.
- The latter are modeled through matrix operators that capture multiple aggregates on graphs, learned using a constrained cross-entropy loss.
- We consider different constraints (including stochasticity, orthogonality and symmetry) acting as regularizers.
- Stochasticity implements random walk Laplacians while orthogonality models multiple aggregation operators with non-overlapping supports; it also avoids redundancy and oversizing the learned GCNs with useless parameters. Symmetry reduces further the number of training parameters.
We introduce a novel framework that learns convolutional filters on graphs together with their topological properties. The latter are modeled through matrix operators that capture multiple aggregates on graphs, learned using a constrained cross-entropy loss.

We consider different constraints (including stochasticity, orthogonality and symmetry) acting as regularizers.

Stochasticity implements random walk Laplacians while orthogonality models multiple aggregation operators with non-overlapping supports; it also avoids redundancy and oversizing the learned GCNs with useless parameters. Symmetry reduces further the number of training parameters.
We introduce a novel framework that learns convolutional filters on graphs together with their topological properties.

The latter are modeled through matrix operators that capture multiple aggregates on graphs, learned using a constrained cross-entropy loss.

We consider different constraints (including stochasticity, orthogonality and symmetry) acting as regularizers.

Stochasticity implements random walk Laplacians while orthogonality models multiple aggregation operators with non-overlapping supports; it also avoids redundancy and oversizing the learned GCNs with useless parameters. Symmetry reduces further the number of training parameters.
Outline

1. Introduction
2. Learning connectivity in GCNs
3. Experiments
4. Conclusion
Considering $G = (\mathcal{V}, \mathcal{E})$ endowed with $\{s(u)\}_u$ and (ii) $A$,

$$(\mathcal{G} \star \mathcal{F})_{\mathcal{V}} = f(A \ U^\top \ W).$$

Here $A U^\top$ acts as a feature extractor which collects non-differential and differential statistics including means and variances of node neighbors, before applying convolutions.

We use the chain rule in order to derive the gradient $\frac{\partial E}{\partial \text{vec}(A)}$ and update $A$ using SGD; we upgrade the latter to learn both the convolutional parameters $W$ together with $A$ while implementing orthogonality, stochasticity and symmetry.

Orthogonality allows us to design $A$ with a minimum number of parameters, stochasticity normalizes nodes by their degrees and allows learning random walk Laplacians, while symmetry reduces further the number of training parameters.
Problem statement

- Considering $G = (V, E)$ endowed with $\{s(u)\}_u$ and (ii) $A$,
  $$(G \ast \mathcal{F})_V = f(A U^\top W).$$

- Here $AU^\top$ acts as a feature extractor which collects non-differential and differential statistics including means and variances of node neighbors, before applying convolutions.

- We use the chain rule in order to derive the gradient $\frac{\partial E}{\partial \text{vec}(A)}$ and update $A$ using SGD; we upgrade the latter to learn both the convolutional parameters $W$ together with $A$ while implementing orthogonality, stochasticity and symmetry.

- Orthogonality allows us to design $A$ with a minimum number of parameters, stochasticity normalizes nodes by their degrees and allows learning random walk Laplacians, while symmetry reduces further the number of training parameters.
Problem statement

- Considering $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ endowed with $\{s(u)\}_{u}$ and (ii) $A$, 
  \[(\mathcal{G} \star \mathcal{F})\mathcal{V} = f(A \mathbf{U}^\top W).\]

- Here $A\mathbf{U}^\top$ acts as a feature extractor which collects non-differential and differential statistics including means and variances of node neighbors, before applying convolutions.

- We use the chain rule in order to derive the gradient $\frac{\partial E}{\partial \text{vec}(A)}$ and update $A$ using SGD; we upgrade the latter to learn both the convolutional parameters $W$ together with $A$ while implementing orthogonality, stochasticity and symmetry.

- Orthogonality allows us to design $A$ with a minimum number of parameters, stochasticity normalizes nodes by their degrees and allows learning random walk Laplacians, while symmetry reduces further the number of training parameters.
Problem statement

- Considering $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ endowed with $\{s(u)\}_u$ and (ii) $A$,

  \[(\mathcal{G} \star \mathcal{F})_{\mathcal{V}} = f(\mathbf{A} \mathbf{U}^\top \mathbf{W}).\]

- Here $\mathbf{A} \mathbf{U}^\top$ acts as a feature extractor which collects non-differential and differential statistics including means and variances of node neighbors, before applying convolutions.

- We use the chain rule in order to derive the gradient $\frac{\partial E}{\partial \text{vec}(\mathbf{A})}$ and update $\mathbf{A}$ using SGD; we upgrade the latter to learn both the convolutional parameters $\mathbf{W}$ together with $\mathbf{A}$ while implementing orthogonality, stochasticity and symmetry.

- Orthogonality allows us to design $\mathbf{A}$ with a minimum number of parameters, stochasticity normalizes nodes by their degrees and allows learning random walk Laplacians, while symmetry reduces further the number of training parameters.
Stochasticity

- Stochasticity requires adding equality and inequality constraints in SGD, i.e., $A_{ij} \in [0, 1]$ and $\sum_q A_{qj} = 1$.
- We consider a reparametrization of the learned matrices, as $A_{ij} = h(\hat{A}_{ij})/\sum_q h(\hat{A}_{qj})$.
- During backpropagation, the gradient of the loss $E$ (now w.r.t $\hat{A}$) is updated using the chain rule as

$$\frac{\partial E}{\partial \hat{A}_{ij}} = \sum_p \frac{\partial E}{\partial A_{pj}} \cdot \frac{\partial A_{pj}}{\partial \hat{A}_{ij}}$$

with

$$\frac{\partial A_{pj}}{\partial \hat{A}_{ij}} = \frac{h'(\hat{A}_{ij})}{\sum_q h(\hat{A}_{qj})} \cdot (\delta_{pi} - A_{pj}).$$

- In practice $h(.) = \exp(.)$ and the new gradient (w.r.t $\hat{A}$) is obtained by multiplying the original one by the Jacobian $J_{stc} = \left[ \frac{\partial A_{pj}}{\partial \hat{A}_{ij}} \right]_{p,i=1}^n$ which merely reduces to $[A_{ij} \cdot (\delta_{pi} - A_{pj})]_{p,i}$.
Stochasticity

- Stochasticity requires adding equality and inequality constraints in SGD, i.e., $A_{ij} \in [0, 1]$ and $\sum_q A_{qj} = 1$.

- We consider a reparametrization of the learned matrices, as $A_{ij} = h(\hat{A}_{ij}) / \sum_q h(\hat{A}_{qj})$.

- During backpropagation, the gradient of the loss $E$ (now w.r.t $\hat{A}$) is updated using the chain rule as

$$\frac{\partial E}{\partial \hat{A}_{ij}} = \sum_p \frac{\partial E}{\partial A_{pj}} \cdot \frac{\partial A_{pj}}{\partial \hat{A}_{ij}}$$

with

$$\frac{\partial A_{pj}}{\partial \hat{A}_{ij}} = \frac{h'(\hat{A}_{ij})}{\sum_q h(\hat{A}_{qj})} \cdot (\delta_{pi} - A_{pj}).$$

- In practice $h(.) = \exp(.)$ and the new gradient (w.r.t $\hat{A}$) is obtained by multiplying the original one by the Jacobian

$$J_{stc} = \left[ \frac{\partial A_{pj}}{\partial \hat{A}_{ij}} \right]_{p,i=1}^n$$

which merely reduces to $[A_{ij} \cdot (\delta_{pi} - A_{pj})]_{p,i}$. 
Stochasticity

- Stochasticity requires adding equality and inequality constraints in SGD, i.e., $A_{ij} \in [0, 1]$ and $\sum_q A_{qj} = 1$.

- We consider a reparametrization of the learned matrices, as $A_{ij} = h(\hat{A}_{ij})/\sum_q h(\hat{A}_{qj})$.

- During backpropagation, the gradient of the loss $E$ (now w.r.t $\hat{A}$) is updated using the chain rule as

$$\frac{\partial E}{\partial \hat{A}_{ij}} = \sum_p \frac{\partial E}{\partial A_{pj}} \cdot \frac{\partial A_{pj}}{\partial \hat{A}_{ij}} \text{ with } \frac{\partial A_{pj}}{\partial \hat{A}_{ij}} = \frac{h'(\hat{A}_{ij})}{\sum_q h(\hat{A}_{qj})} \cdot (\delta_{pi} - A_{pj}).$$

- In practice $h(.) = \exp(.)$ and the new gradient (w.r.t $\hat{A}$) is obtained by multiplying the original one by the Jacobian $J_{stc} = \left[ \frac{\partial A_{pj}}{\partial \hat{A}_{ij}} \right]_{p,i=1}^n$ which merely reduces to $[A_{ij} \cdot (\delta_{pi} - A_{pj})]_{p,i}$. 

Hichem SAHBI

Learning Connectivity with Graph Convolutional Networks
Stochasticity

- Stochasticity requires adding equality and inequality constraints in SGD, i.e., $A_{ij} \in [0, 1]$ and $\sum_q A_{qj} = 1$.
- We consider a reparametrization of the learned matrices, as $A_{ij} = h(\hat{A}_{ij})/\sum_q h(\hat{A}_{qj})$.
- During backpropagation, the gradient of the loss $E$ (now w.r.t $\hat{A}$) is updated using the chain rule as

$$\frac{\partial E}{\partial \hat{A}_{ij}} = \sum_p \frac{\partial E}{\partial A_{pj}} \cdot \frac{\partial A_{pj}}{\partial \hat{A}_{ij}} \quad \text{with} \quad \frac{\partial A_{pj}}{\partial \hat{A}_{ij}} = \frac{h'(\hat{A}_{ij})}{\sum_q h(\hat{A}_{qj})} \cdot (\delta_{pi} - A_{pj}).$$

- In practice $h(.) = \exp(.)$ and the new gradient (w.r.t $\hat{A}$) is obtained by multiplying the original one by the Jacobian $J_{stc} = \left[ \frac{\partial A_{pj}}{\partial \hat{A}_{ij}} \right]_{p,i=1}^n$ which merely reduces to $[A_{ij} \cdot (\delta_{pi} - A_{pj})]_{p,i}$. 

Orthogonality

- Learning multiple \( \{A_k\}_k \) allows us to capture different **graph topologies** when achieving aggregation and convolution.

\[
(G \star F)_{ij} = f \left( \sum_{k=1}^{K} A_k U^\top W_k \right)
\]

\[
\min \{A_k\}_k, W \quad \text{s.t.} \quad E(A_1, \ldots, A_K; W) \\
A_k \odot A_k > 0_n, \quad A_k \odot A_{k'} = 0_n \quad \forall k, k' \neq k.
\]

- We consider \( \exp(\gamma \hat{A}_k) \odot (\sum_{r=1}^{K} \exp(\gamma \hat{A}_r)) \) as a softmax reparametrization of \( A_k \), with \( \{\hat{A}_k\}_k \) free parameters in \( \mathbb{R}^{n \times n} \).
- By choosing a large value of \( \gamma \), it becomes possible to implement \( \epsilon \)-orthogonality; a surrogate property where only one entry \( A_{kij} \gg 0 \) while all others \( \{A_{k'ij}\}_{k' \neq k} \) vanish.
- The setting of \( \gamma \) and updated Jacobians are in the paper.
Orthogonality

- Learning multiple $\{A_k\}_k$ allows us to capture different graph topologies when achieving aggregation and convolution.

$$ (G \star \mathcal{F})_v = f\left( \sum_{k=1}^{K} A_k U^\top W_k \right) $$

$$ \min_{\{A_k\}_k, W} \quad E(A_1, \ldots, A_K; W) $$

s.t. \quad $$ A_k \odot A_k > 0_n, \quad A_k \odot A_{k'} = 0_n \quad \forall k, k' \neq k. $$

- We consider $\exp(\gamma \hat{A}_k) \odot (\sum_{r=1}^{K} \exp(\gamma \hat{A}_r))$ as a softmax reparametrization of $A_k$, with $\{\hat{A}_k\}_k$ free parameters in $\mathbb{R}^{n \times n}$.

- By choosing a large value of $\gamma$, it becomes possible to implement $\epsilon$-orthogonality; a surrogate property where only one entry $A_{kij} \gg 0$ while all others $\{A_{k'ij}\}_{k' \neq k}$ vanish.

- The setting of $\gamma$ and updated Jacobians are in the paper.
Orthogonality

- Learning multiple \( \{A_k\}_k \) allows us to capture different graph topologies when achieving aggregation and convolution.

\[
(G \ast F)_V = f \left( \sum_{k=1}^{K} A_k U^\top W_k \right)
\]

\[
\min_{\{A_k\}_k, W} E(A_1, \ldots, A_K; W)
\text{ s.t. } A_k \odot A_k > 0_n, A_k \odot A_{k'} = 0_n \quad \forall k, k' \neq k.
\]

- We consider \( \exp(\gamma \hat{A}_k) \odot (\sum_{r=1}^{K} \exp(\gamma \hat{A}_r)) \) as a softmax reparametrization of \( A_k \), with \( \{\hat{A}_k\}_k \) free parameters in \( \mathbb{R}^{n \times n} \).

- By choosing a large value of \( \gamma \), it becomes possible to implement \( \epsilon \)-orthogonality; a surrogate property where only one entry \( A_{kij} \gg 0 \) while all others \( \{A_{k'ij}\}_{k' \neq k} \) vanish.

- The setting of \( \gamma \) and updated Jacobians are in the paper.
Learning multiple \( \{ \mathbf{A}_k \}_k \) allows us to capture different **graph topologies** when achieving aggregation and convolution.

\[
(G \star F)_\nu = f \left( \sum_{k=1}^{K} \mathbf{A}_k \mathbf{U}^\top \mathbf{W}_k \right)
\]

\[
\min_{\{ \mathbf{A}_k \}_k, \mathbf{W}} \quad E(\mathbf{A}_1, \ldots, \mathbf{A}_K; \mathbf{W})
\]
\[
\text{s.t.} \quad \mathbf{A}_k \odot \mathbf{A}_k > \mathbf{0}_n, \quad \mathbf{A}_k \odot \mathbf{A}_{k'} = \mathbf{0}_n \quad \forall k, k' \neq k.
\]

We consider \( \exp(\gamma \hat{\mathbf{A}}_k) \odot (\sum_{r=1}^{K} \exp(\gamma \hat{\mathbf{A}}_r)) \) as a **softmax reparametrization** of \( \mathbf{A}_k \), with \( \{ \hat{\mathbf{A}}_k \}_k \) free parameters in \( \mathbb{R}^{n \times n} \).

By choosing a large value of \( \gamma \), it becomes possible to implement \( \epsilon \)-**orthogonality**; a surrogate property where only one entry \( \mathbf{A}_{kij} \gg 0 \) while all others \( \{ \mathbf{A}_{k'ij} \}_{k' \neq k} \) vanish.

The setting of \( \gamma \) and updated Jacobians are in the paper.
Symmetry and combination

- Symmetry is guaranteed by considering the reparametrization of each matrix as $A_k = \frac{1}{2}(\hat{A}_k + \hat{A}_k^T)$ with $\hat{A}_k$ being free.
- Symmetry is maintained by multiplying the original gradient by the Jacobian

$$J_{\text{sym}} = \frac{1}{2} \left[ 1\{k=k'\} \cdot 1\{(i'=j', j'=i') \lor (i'=j', j'=i')\} \right]_{ijk, i'j'k'}.$$

- One may combine symmetry with all the aforementioned constraints by multiplying the underlying Jacobians, so the final gradient is obtained by multiplying the original one as

$$\frac{\partial E}{\partial \text{vec}(\{\hat{A}_k\}_k)} = J_{(\text{sym or stc})} \cdot J_{\text{orth}} \cdot \frac{\partial E}{\partial \text{vec}(\{A_k\}_k)}.$$
Symmetry and combination

- Symmetry is guaranteed by considering the reparametrization of each matrix as $A_k = \frac{1}{2}(\hat{A}_k + \hat{A}_k^\top)$ with $\hat{A}_k$ being free.

- Symmetry is maintained by multiplying the original gradient by the Jacobian

$$J_{\text{sym}} = \frac{1}{2} \left[ 1 \{ k=k' \} \cdot 1 \{ (i=i', j=j') \vee (i=j', j=i') \} \right]_{ijk,i'j'k'}$$

- One may combine symmetry with all the aforementioned constraints by multiplying the underlying Jacobians, so the final gradient is obtained by multiplying the original one as

$$\frac{\partial E}{\partial \text{vec}(\{\hat{A}_k\}_k)} = J_{(\text{sym or stc})} \cdot J_{\text{orth}} \cdot \frac{\partial E}{\partial \text{vec}(\{A_k\}_k)}.$$
Symmetry and combination

- Symmetry is guaranteed by considering the reparametrization of each matrix as $\mathbf{A}_k = \frac{1}{2}(\mathbf{\hat{A}}_k + \mathbf{\hat{A}}_k^\top)$ with $\mathbf{\hat{A}}_k$ being free.
- Symmetry is maintained by multiplying the original gradient $\frac{\partial E}{\partial \text{vec}(\{\mathbf{A}_k\}_k)}$ by the Jacobian

$$
\mathbf{J}_{\text{sym}} = \frac{1}{2} \left[ 1\{k=k'\} \cdot 1\{(i=i', j=j') \lor (i=j', j=i')\} \right]_{ijk, i'j'k'}.
$$

- One may combine symmetry with all the aforementioned constraints by multiplying the underlying Jacobians, so the final gradient is obtained by multiplying the original one as

$$
\frac{\partial E}{\partial \text{vec}(\{\mathbf{\hat{A}}_k\}_k)} = \mathbf{J}_{(\text{sym or stc})} \cdot \mathbf{J}_{\text{orth}} \cdot \frac{\partial E}{\partial \text{vec}(\{\mathbf{A}_k\}_k)}.
$$
2 Learning connectivity in GCNs

3 Experiments

4 Conclusion
We evaluate our GCN on the task of action recognition, using the SBU Kinect dataset.

This is an interaction dataset acquired using the Microsoft Kinect sensor; it includes in total 282 video sequences belonging to $C = 8$ categories with variable duration, viewpoint changes and interacting individuals.

In all these experiments, we use the same evaluation protocol as the one suggested in (SBU12) (i.e., train-test split) and we report the average accuracy over all the classes of actions.

We trained our GCNs for 3000 epochs, with a batch size of 200, a momentum of 0.9 and a learning rate $\nu$ that decreases as $\nu \leftarrow \nu \times 0.99$ (resp. increases as $\nu \leftarrow \nu / 0.99$).
Input skeleton graphs

Motion trajectory (v)
(raw coordinates)

Temporal Chunking
## Performances

<table>
<thead>
<tr>
<th>Oper</th>
<th>Const</th>
<th>none</th>
<th>sym</th>
<th>orth</th>
<th>stc</th>
<th>sym+orth</th>
<th>orth+stc</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>HPM.</td>
<td>$K = 1$</td>
<td>89.2308</td>
<td>92.3077</td>
<td>–</td>
<td>89.2308</td>
<td>–</td>
<td>–</td>
<td>90.2564</td>
</tr>
<tr>
<td></td>
<td>$K = 4$</td>
<td>87.6923</td>
<td>89.2308</td>
<td>89.2308</td>
<td>87.6923</td>
<td>90.7692</td>
<td>92.3077</td>
<td>89.4872</td>
</tr>
<tr>
<td></td>
<td>$K = 8$</td>
<td>90.7692</td>
<td>95.3846</td>
<td>92.3077</td>
<td>90.7692</td>
<td>92.3077</td>
<td>92.3077</td>
<td>90.7692</td>
</tr>
<tr>
<td></td>
<td>Mean</td>
<td>89.2308</td>
<td>92.3077</td>
<td>90.7692</td>
<td>89.2308</td>
<td>91.5384</td>
<td>92.3077</td>
<td>90.7692</td>
</tr>
<tr>
<td>LPM.</td>
<td>$K = 1$</td>
<td>92.3077</td>
<td>87.6923</td>
<td>–</td>
<td>95.3846</td>
<td>–</td>
<td>–</td>
<td>91.7949</td>
</tr>
<tr>
<td></td>
<td>$K = 4$</td>
<td>92.3077</td>
<td>92.3077</td>
<td>93.8462</td>
<td>95.3846</td>
<td>90.7692</td>
<td>96.9231</td>
<td>93.5897</td>
</tr>
<tr>
<td></td>
<td>$K = 8$</td>
<td>95.3846</td>
<td>90.7692</td>
<td>87.6923</td>
<td>93.8462</td>
<td>93.8462</td>
<td>93.8462</td>
<td>92.3077</td>
</tr>
<tr>
<td></td>
<td>Mean</td>
<td>93.3333</td>
<td>90.2564</td>
<td>90.7692</td>
<td>94.8718</td>
<td>92.3077</td>
<td>94.6154</td>
<td>92.7180</td>
</tr>
<tr>
<td>Our</td>
<td>$K = 1$</td>
<td>95.3846</td>
<td>93.8462</td>
<td>–</td>
<td>95.3846</td>
<td>–</td>
<td>–</td>
<td>94.8718</td>
</tr>
<tr>
<td></td>
<td>$K = 4$</td>
<td>93.8462</td>
<td>95.3846</td>
<td>95.3846</td>
<td>96.9231</td>
<td>93.8462</td>
<td>98.4615</td>
<td>95.6410</td>
</tr>
<tr>
<td></td>
<td>$K = 8$</td>
<td>92.3077</td>
<td>93.8462</td>
<td>95.3846</td>
<td>90.7692</td>
<td>95.3846</td>
<td>90.7692</td>
<td>93.0769</td>
</tr>
<tr>
<td></td>
<td>Mean</td>
<td>93.8462</td>
<td>94.3590</td>
<td>95.3846</td>
<td>94.3590</td>
<td>94.6154</td>
<td>94.6154</td>
<td>94.4615</td>
</tr>
</tbody>
</table>

Learned connectivity (examples)
## Comparison

<table>
<thead>
<tr>
<th>Methods</th>
<th>Perfs</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCNConv [57]</td>
<td>90.00</td>
</tr>
<tr>
<td>ArmaConv [61]</td>
<td>96.00</td>
</tr>
<tr>
<td>SGCConv [59]</td>
<td>94.00</td>
</tr>
<tr>
<td>ChebyNet [58]</td>
<td>96.00</td>
</tr>
<tr>
<td>Raw coordinates [53]</td>
<td>49.7</td>
</tr>
<tr>
<td>Joint features [53]</td>
<td>80.3</td>
</tr>
<tr>
<td>Interact Pose [62]</td>
<td>86.9</td>
</tr>
<tr>
<td>CHARM [63]</td>
<td>83.9</td>
</tr>
<tr>
<td>HBRNN-L [64]</td>
<td>80.35</td>
</tr>
<tr>
<td>Co-occurrence LSTM [66]</td>
<td>90.41</td>
</tr>
<tr>
<td>ST-LSTM [67]</td>
<td>93.3</td>
</tr>
<tr>
<td>Topological pose ordering [70]</td>
<td>90.5</td>
</tr>
<tr>
<td>STA-LSTM [56]</td>
<td>91.51</td>
</tr>
<tr>
<td>GCA-LSTM [55]</td>
<td>94.9</td>
</tr>
<tr>
<td>VA-LSTM [68]</td>
<td>97.2</td>
</tr>
<tr>
<td>DeepGRU [54]</td>
<td>95.7</td>
</tr>
<tr>
<td>Riemannian manifold trajectory [69]</td>
<td>93.7</td>
</tr>
<tr>
<td>Our best GCN model</td>
<td>98.46</td>
</tr>
</tbody>
</table>
Outline

1. Introduction
2. Learning connectivity in GCNs
3. Experiments
4. Conclusion
We introduce in this paper a novel method which learns connectivity that "optimally" defines the support of aggregations and convolutions in GCNs.

We investigate different settings which allow extracting non-differential and differential features as well as their combination before applying convolutions.

We also consider different constraints (including orthogonality and stochasticity) which act as regularizers on the learned matrix operators.

Experiments conducted on the challenging task of skeleton-based action recognition show the clear gain of the proposed method w.r.t different baselines as well as the related work.