HAL open science

Lightweight Connectivity In Graph Convolutional Networks For Skeleton-Based Recognition

Hichem Sahbi

▶ To cite this version:

Hichem Sahbi. Lightweight Connectivity In Graph Convolutional Networks For Skeleton-Based Recognition. IEEE International Conference on Image Processing (ICIP), Sep 2021, Anchorage, AK (virtual), United States. pp.2329-2333, 10.1109/ICIP42928.2021.9506774 . hal-03431268

HAL Id: hal-03431268

https://hal.science/hal-03431268v1

Submitted on 16 Nov 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

LIGHTWEIGHT CONNECTIVITY IN GRAPH CONVOLUTIONAL NETWORKS FOR SKELETON-BASED RECOGNITION

Hichem Sahbi

Sorbonne University, UPMC, CNRS, LIP6, F-75005 Paris, France

ABSTRACT

Graph convolutional networks (GCNs) aim at extending deep learning to arbitrary irregular domains, namely graphs. Their success is highly dependent on how the topology of input graphs is defined and most of the existing GCN architectures rely on predefined or handcrafted graph structures.
In this paper, we introduce a novel method that learns the topology (or connectivity) of input graphs as a part of GCN design. The main contribution of our method resides in building an orthogonal connectivity basis that optimally aggregates nodes, through their neighborhood, prior to achieve convolution. Our method also considers a stochasticity criterion which acts as a regularizer that makes the learned basis and the underlying GCNs lightweight while still being highly effective. Experiments conducted on the challenging task of skeleton-based hand-gesture recognition show the high effectiveness of the learned GCNs w.r.t. the related work.

***Index Terms*—** Graph convolutional networks, lightweight connectivity design, skeleton-based recognition.

1. INTRODUCTION

Deep learning is currently witnessing a major interest in different fields including image processing and pattern recognition [1, 2]. Its principle consists in learning multi-layered convolutional, pooling and fully connected operations that extract representations which capture low, mid and high-level characteristics of patterns while maximizing their classification performances. Most of the existing deep learning architectures [3–11] are targeted to vectorial data; i.e., data sitting on top of regular domains including images. However, other data require extending deep learning to irregular domains (namely graphs [13–15]) such as skeletons in action recognition. While convolutional operations on regular domains are well defined, their extension to irregular ones (i.e., graphs) is generally ill-posed and remains a major challenge.

Two categories of GCNs exist in the literature, spatial and spectral [12]. Spatial methods achieve node aggregations prior to apply convolutions using inner products while spectral techniques rely on the well defined graph Fourier transform. Whereas spatial methods are known to be effective compared to spectral ones, their success is highly dependent on the topology of input graphs, and most of the existing solutions rely on handcrafted or predefined graph structures. The latters are based on similarities or inherent properties of the targeted applications [13–15, 21, 30, 35, 40] (e.g., node relationships in social networks, edges in 3D modeling, etc). These structures are usually powerless to capture the most prominent relationships between nodes as their design is agnostic to the targeted application. For instance, when considering node relationships in skeletons, these links capture the anthropometric characteristics of individuals which are useful for their identification, while other connections, yet to infer, are important for recognizing their actions. Hence, in spite of being relatively effective, the potential of these GCN methods is not fully explored as the setting of their graphs is either oblivious to the tasks at hand or achieved using the tedious cross validation.

Graph inference is generally ill-posed, NP-hard [16] and most of the existing approaches rely on constraints (similarity, smoothness, sparsity, band-limitedness) for its conditioning [17–20, 22]. Particularly in GCNs, recent advances aim at defining graph topology that best fits a given task [23–26]. For instance, [23] proposes a graph network for semi-supervised classification that learns graph topology with sparse structure given a cloud of points; node-to-node connections are modeled with a joint probability distribution on Bernoulli random variables whose parameters are found using bi-level optimization. A computationally more efficient variant is introduced in [24] using a weighted cosine similarity and edge thresholding. Other solutions make improvement w.r.t. the original GCNs [15] by exploiting symmetric matrices [25] and discovering hidden structural relations (unspecified in the original graphs) using a so-called residual graph adjacency matrix, and by learning a distance function over nodes. The work in [26] introduces a dual architecture with two parallel graph convolutional layers sharing the same parameters, and considers a normalized adjacency and a positive point-wise mutual information matrix to capture node co-occurrences through random walks sampled from graphs.

In this paper, we introduce a novel framework that designs graphs as a part of end-to-end GCN learning. Our design principle is based on the minimization of a constrained loss whose solution corresponds not only to the convolutional parameters of GCNs but also the underlying adjacency matrices

that capture the topology of input graphs. Our contribution in this paper differs from the aforementioned related work in multiple aspects. On the one hand, in contrast to many existing methods – e.g., [27] which consider a single adjacency matrix shared through power series – the matrix operators designed in our contribution are non-parametrically learned and this provides more flexibility to our design. On the other hand, constraining these matrices, through orthogonality and stochasticity, allows achieving structured regularization. This mitigates overfitting and allows learning lightweight GCN architectures. In contrast to unstructured lightweight network design (e.g., magnitude pruning), our proposed method captures (through orthogonality and stochasticity) the structural relationships between parameters in the learned GCNs. It also maintains completeness and minimality of the learned representations by finding the most discriminating and lightweight GCNs as supported in our experiments.

2. LEARNING LIGHTWEIGHT CONNECTIVITY

Let $\mathcal{S} = \{\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)\}_i$ denote a collection of graphs with $\mathcal{V}_i$, $\mathcal{E}_i$ being respectively the nodes and the edges of $\mathcal{G}_i$. Each graph $\mathcal{G}_i$ (denoted for short as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$) is endowed with a signal $\{\psi(u) \in \mathbb{R}^s : u \in \mathcal{V}\}$ and associated with an adjacency matrix $\mathbf{A}$ with each entry $\mathbf{A}_{uu'} > 0$ iff $(u, u') \in \mathcal{E}$ and 0 otherwise. GCNs aim at learning a set of filters $\mathcal{F} = \{g_\theta = (\mathcal{V}_\theta, \mathcal{E}_\theta)\}_{\theta=1}^{C}$ that define convolution on n nodes of $\mathcal{G}$ (with $n = |\mathcal{V}|$) as

$$(\mathcal{G} \star \mathcal{F})_\mathcal{V} = f\big(\mathbf{A}\,\mathbf{U}^\top\,\mathbf{W}\big), \tag{1}$$

here $^\top$ stands for transpose, $\mathbf{U} \in \mathbb{R}^{s\times n}$ is the graph signal, $\mathbf{W} \in \mathbb{R}^{s\times C}$ is the matrix of convolutional parameters corresponding to the C filters and $f(.)$ is a nonlinear activation applied entrywise. In Eq. 1, the input signal $\mathbf{U}$ is projected using $\mathbf{A}$ and this provides for each node u, the aggregate set of its neighbors. When $\mathbf{A}$ is common to all graphs[1], entries of $\mathbf{A}$ could be handcrafted or learned so Eq. (1) implements a convolutional block with two layers; the first one aggregates signals in $\mathcal{N}(\mathcal{V})$ (sets of node neighbors) by multiplying $\mathbf{U}$ with $\mathbf{A}$ while the second layer achieves convolution by multiplying the resulting aggregates with the C filters in $\mathbf{W}$.

2.1. Orthogonality-driven connectivity

Learning multiple adjacency matrices (denoted as $\{\mathbf{A}_k\}_{k=1}^{K}$) allows us to capture different contexts and graph topologies when achieving aggregation and convolution. With multiple matrices $\{\mathbf{A}_k\}_k$ (and associated convolutional filter parameters $\{\mathbf{W}_k\}_k$), Eq. 1 is updated as

$$(\mathcal{G} \star \mathcal{F})_\mathcal{V} = f\bigg(\sum_{k=1}^{K} \mathbf{A}_k \mathbf{U}^\top \mathbf{W}_k\bigg). \tag{2}$$

[1]e.g., when considering a common graph structure for all actions in videos.

If aggregation produces, for a given $u \in \mathcal{V}$, linearly dependent vectors $\mathcal{X}_u = \{\sum_{u'} \mathbf{A}_{kuu'}.\psi(u')\}_k$, then convolution will also generate linearly dependent representations with an overestimated number of training parameters in the null space of $\mathcal{X}_u$. Besides, the tensor $\{\mathbf{A}_k\}_k$ used for aggregation, may also generate overlapping and redundant contexts.
Provided that $\{\psi(u')\}_{u'\in\mathcal{N}_r(u)}$ are linearly independent, the sufficient condition that makes vectors in $\mathcal{X}_u$ linearly independent reduces to constraining $(\mathbf{A}_{kuu'})_{k,u'}$ to lie on the Stiefel manifold (see for instance [28]) defined as $V_K(\mathbb{R}^n) = \{\mathbf{M} \in \mathbb{R}^{K\times n} : \mathbf{M}\,\mathbf{M}^\top = \mathbf{I}_K\}$ (with $\mathbf{I}_K$ being the $K \times K$ identity matrix) which thereby guarantees orthonormality and minimality of $\{\mathbf{A}_1, \ldots, \mathbf{A}_K\}$[2]. A less compelling condition is orthogonality, i.e., $\langle \mathbf{A}_k, \mathbf{A}_{k'}\rangle_F = 0$ and $\mathbf{A}_k \geq \mathbf{0}_{n\times n}$, $\mathbf{A}_{k'} \geq \mathbf{0}_{n\times n}$, $\forall k \neq k'$ — with $\langle,\rangle_F$ being the Hilbert-Schmidt (or Frobenius) inner product defined as $\langle \mathbf{A}_k, \mathbf{A}_{k'}\rangle_F = \mathbf{Tr}(\mathbf{A}_k^\top \mathbf{A}_{k'})$ — and this equates $\mathbf{A}_k \odot \mathbf{A}_{k'} = \mathbf{0}_{n\times n}$, $\forall k \neq k'$, with $\odot$ denoting the entrywise Hadamard product and $\mathbf{0}_{n\times n}$ the $n \times n$ null matrix.

Considering orthogonality (as discussed above), the tensor $\{\mathbf{A}_k\}_k$ and $\mathbf{W} = \{\mathbf{W}_k\}_k$ are learned as

$$\begin{array}{lll} \min_{\{\mathbf{A}_k\geq 0\}_k, \mathbf{W}} & E\big(\mathbf{A}_1, \ldots, \mathbf{A}_K; \mathbf{W}\big) & \\ \text{s.t.} & \mathbf{A}_k \odot \mathbf{A}_k > \mathbf{0}_{n\times n} & \\ & \mathbf{A}_k \odot \mathbf{A}_{k'} = \mathbf{0}_{n\times n} & \forall k, k' \neq k \\ & \mathbf{1}_n^\top \mathbf{A}_k = \mathbf{1}_n^\top. & \end{array} \tag{3}$$

being E the cross entropy loss and $\mathbf{1}_n^\top$ a vector of n ones. In the above minimization problem, the first and the second constraints correspond to orthogonality while the third one to column-stochasticity. The latter is added in order to ensure that all of the entries in $\mathbf{A}_k$ are positive and each column sums to one; i.e., each matrix $\mathbf{A}_k$ models a Markov chain whose i-th row and j-th column entry provides the probability of transition from one node u_j to u_i in $\mathcal{G}$. Note that orthogonality (as designed subsequently) allows learning sparse adjacency matrices while column-stochasticity provides extra sparsity; it acts as a structured regularizer that enhances further the generalization power of the learned GCNs[3].

2.2. Optimization

A natural approach to solve Eq. (3) is to iteratively and alternately minimize over one matrix while keeping all the others fixed. However — and besides the non-convexity of the loss — the feasible set formed by these $O(K^2)$ bi-linear constraints is not convex w.r.t. $\{\mathbf{A}_k\}_k$. Moreover, this iterative procedure is computationally expensive as it requires solving multiple instances of constrained projected gradient descent and the number of necessary iterations to reach convergence

[2]Note that K should not exceed the rank of $\{\psi(u')\}_{u'\in\mathcal{N}_r(u)}$ which is upper bounded by $\min(|\mathcal{V}|, s)$; s is again the dimension of the graph signal.

[3]Without stochasticity, one has to consider a normalization layer (with extra parameters), especially on graphs with heterogeneous degrees in order to reduce the covariate shift and distribute the transition probability evenly through nodes before convolutions.

is large in practice. All these issues make solving this problem challenging and computationally intractable even for reasonable values of K and n. In what follows, we investigate a workaround that optimizes these matrices while guaranteeing their orthogonality and stochasticity as a part of optimization.

Orthogonality. Let $\exp(\gamma\hat{\mathbf{A}}_k) \oslash (\sum_{r=1}^{K}\exp(\gamma\hat{\mathbf{A}}_r))$ be a softmax reparametrization of $\mathbf{A}_k$, with $\oslash$ being the entrywise Hadamard division and $\{\hat{\mathbf{A}}_k\}_k$ free parameters in $\mathbb{R}^{n\times n}$. It becomes possible to implement orthogonality by choosing large values of γ to make this softmax *crisp*; i.e., only one entry $\mathbf{A}_{kij} \gg 0$ while all others $\{\mathbf{A}_{k'ij}\}_{k'\neq k}$ vanishing thereby leading to $\mathbf{A}_k \odot \mathbf{A}_{k'} = \mathbf{0}_{n\times n}, \forall k, k' \neq k$. By plugging this *crispmax* reparametrization into Eq. 3, the gradient of the loss E (now w.r.t. $\{\hat{\mathbf{A}}_k\}_k$) is updated using the chain rule as

$$\frac{\partial E}{\partial \mathbf{vec}(\{\hat{\mathbf{A}}_k\}_k)} = \mathbf{J}_{\text{orth}}.\frac{\partial E}{\partial \mathbf{vec}(\{\mathbf{A}_k\}_k)}, \tag{4}$$

being $\mathbf{vec}(\{\mathbf{A}_k\}_k)$ a vectorization of $\{\mathbf{A}_k\}_k$ and $(\mathbf{i},\mathbf{j}) = (kij, k'i'j')$ an entry of the Jacobian $\mathbf{J}_{\text{orth}}$ as

$$\begin{cases} \gamma\mathbf{A}_{kij}.(1-\mathbf{A}_{kij}) & \text{if } k=k', i=i', j=j' \\ -\gamma\mathbf{A}_{kij}.\mathbf{A}_{k'ij} & \text{if } k\neq k', i=i', j=j' \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

here $\frac{\partial E}{\partial \mathbf{vec}(\{\mathbf{A}_k\}_k)}$ is obtained from layerwise gradient back-propagation. However, with this reparametrization, large values of γ may lead to numerical instability when evaluating the exponential. We circumvent this by choosing γ that satisfies ϵ-orthogonality: a surrogate property defined subsequently.

Definition 1 (ϵ-orthogonality) *A basis $\{\mathbf{A}_k\}_k$ is ϵ-orthogonal if $\mathbf{A}_k \odot \mathbf{A}_{k'} \leq \epsilon\, \mathbb{1}_{n\times n}$, $\forall k, k' \neq k$, with $\mathbb{1}_{n\times n}$ being the $n\times n$ unitary matrix.*

Considering the above definition, (nonzero) matrices belonging to an ϵ-orthogonal basis are linearly independent w.r.t. $\langle .,. \rangle_F$ (provided that γ is sufficiently large) and hence this basis is also minimal. The following proposition provides a tight lower bound on γ that satisfies ϵ-orthogonality.

Proposition 1 (ϵ-orthogonality bound) *Consider $\{\mathbf{A}_{kij}\}_{ij}$ as the entries of the crispmax reparametrized matrix $\mathbf{A}_k$ defined as $\exp(\gamma\hat{\mathbf{A}}_k) \oslash \big(\sum_{r=1}^{K}\exp(\gamma\hat{\mathbf{A}}_r)\big)$. Provided that $\exists\delta > 0 : \forall i, j, \ell', \exists!\ell,\ \hat{\mathbf{A}}_{\ell ij} \geq \hat{\mathbf{A}}_{\ell' ij} + \delta$ (with $\ell' \neq \ell$) and if γ is at least*

$$\frac{1}{\delta}\ln\left(\frac{K\sqrt{(1-2\epsilon)}}{1-\sqrt{(1-2\epsilon)}}+1\right)$$

then $\{\mathbf{A}_1, \ldots, \mathbf{A}_K\}$ is ϵ-orthogonal.

In view of space limitation, details of the proof are omitted and may be found in [46]. Following the above proposition, setting γ to the above lower bound guarantees ϵ-orthogonality. For instance, when $K = 2$, $\delta = 0.01$ and provided that $\gamma \geq 530$, one may obtain 0.01-orthogonality which is almost a strict orthogonality. This property is satisfied as long as one slightly disrupts the entries of $\{\hat{\mathbf{A}}_k\}_k$ with random noise during training[4]. However, this may still lead to another limitation; precisely, bad local minima are observed due to an *early* convergence to crisp adjacency matrices. We prevent this by steadily annealing the temperature $1/\gamma$ of the softmax through training epochs (using $\frac{\gamma.\text{epoch}}{\text{max_epochs}}$ instead of γ). This focuses the optimization initially on the loss, and as training evolves, the temperature cools down and allows reaching the aforementioned lower bound (thereby crispmax) and ϵ-orthogonality at convergence.

[4] whatever the range of entries in these matrices $\{\hat{\mathbf{A}}_k\}_k$.

Lightweight connectivity with stochasticity. Unless explicitly mentioned, $\mathbf{A}_k$ is simply rewritten as $\mathbf{A}$. We consider a reparametrization $\mathbf{A} = h(\hat{\mathbf{A}})\mathbf{D}(h(\hat{\mathbf{A}}^\top))^{-1}$, with $\mathbf{D}(.)$ being the degree matrix operator, h a strictly monotonic positive function and this allows a free setting of the matrix $\hat{\mathbf{A}}$ during optimization while guaranteeing stochasticity. In practice, h is set to $\exp$ and the original gradient is obtained, similarly to Eq. 4, from layerwise gradient back propagation by multiplying the original gradient by the Jacobian $[\mathbf{J}_{\text{stc}}]_{ij,i'j'} = [\mathbf{A}_{i'j'}.(\delta_{ii'} - \mathbf{A}_{ij})]$ with $\delta_{ii'} = 1_{\{i=i'\}}$. Note that stochasticity, when combined with orthogonality, leightens connectivity by a factor n compared to orthogonality whose factor does not exceed K; this combination is obtained by multiplying the underlying Jacobians, so the final gradient becomes

$$\frac{\partial E}{\partial \mathbf{vec}(\{\hat{\mathbf{A}}_k\}_k)} = \mathbf{J}_{\text{stc}}.\mathbf{J}_{\text{orth}}.\frac{\partial E}{\partial \mathbf{vec}(\{\mathbf{A}_k\}_k)}, \tag{6}$$

and this order of application is strict, as orthogonality sustains after stochasticity while the converse is not necessarily guaranteed at the end of the optimization process.

3. EXPERIMENTS

Database and settings. We evaluate the performance of our GCN on the task of action recognition using the First-Person Hand Action (FPHA) dataset [29]. The latter includes 1175 skeletons belonging to 45 action categories which are performed by 6 different individuals in 3 scenarios. Action categories are highly variable with inter and intra subject variability including style, speed, scale and viewpoint. Each video (sequence of skeletons) is initially described with a handcrafted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where each node $v_j \in \mathcal{V}$ corresponds to the j-th hand-joint trajectory (denoted as $\{\hat{p}_j^t\}_t$) and an edge $(v_j, v_i) \in \mathcal{E}$ exists iff the j-th and the i-th trajectories are spatially connected. Each trajectory in $\mathcal{G}$ is processed using *temporal chunking*: first, the total duration of a sequence is split into M equally-sized temporal chunks ($M = 4$ in practice), then the trajectory coordinates $\{\hat{p}_j^t\}_t$ are assigned to the M chunks (depending on their time stamps) prior to concatenate the averages of these chunks. This produces the raw description of v_j, again denoted as $\psi(v_j)$.

Implementation details. We trained the GCNs end-to-end using the Adam optimizer for 2,800 epochs with a batch size equal to 600, a momentum of 0.9 and a global learning rate (denoted as $\nu(t)$) inversely proportional to the speed of

change of the cross entropy loss used to train our networks. When this speed increases (resp. decreases), $\nu(t)$ decreases as $\nu(t) \leftarrow \nu(t-1) \times 0.99$ (resp. increases as $\nu(t) \leftarrow \nu(t-1)/0.99$). In all these experiments, we use a GeForce GTX 1070 GPU device (with 8 GB memory). We evaluate the performances using the 1:1 setting proposed in [29] with 600 action sequences for training and 575 for testing, and we report the average accuracy over all the classes of actions.

Performances and comparison. We compare the performances of our GCN design against two baselines: handcrafted and learned. In the first baseline (known as power map), all the matrices $\{\mathbf{A}_k\}_k$ are evaluated upon the adjacency matrix $\mathbf{A}$ (taken from the input skeletons) as $\mathbf{A}_k = \mathbf{A}^{(k)}$ with $\mathbf{A}^{(k)} = \mathbf{A}^{(k-1)}\mathbf{A}$, $\mathbf{A}^{(0)} = \mathbf{I}$ and this defines nested supports for convolutions. In the second baseline, all the adjacency matrices $\{\mathbf{A}_k\}_k$ are learned using the objective function (3) but w/o orthogonality and stochasticity constraints. Table 1 shows a comparison with these baselines and an ablation study of our complete model and the impact of orthogonality (separately and combined) on the performances. These results show that orthogonality has a clear and a consistent positive impact on the performances while stochasticity (when combined with orthogonality) provides lightweight GCNs with an extra gain in accuracy. Clearly, these two constraints act as regularizers that also reduce the number of training parameters thereby leading to highly effective and also efficient GCNs. In order to further investigate the impact of these two constraints, we compare the underlying GCNs against lightweight ones obtained differently, with magnitude pruning; the latter consists first in zeroing the smallest parameters in the learned GCNs, and then fine-tuning the remaining parameters. As shown in table 1, lightweight GCNs, trained with orthogonality and stochasticity, clearly outperform those obtained with magnitude pruning+fine-tuning. Finally, we compare the classification performances of our GCN against other related methods in action recognition ranging from sequence based such as LSTM to deep graph (non-vectorial) methods, etc. (see table 2 and references within). From the results in these tables, our GCN brings a noticeable gain w.r.t. related state of the art methods.

4. CONCLUSION

We introduce in this paper a novel framework that designs graph topology as a part of an "end-to-end" GCN learning. This topology is captured using multiple adjacency matrices whose optimization is constrained with orthogonality and stochasticity. The former makes it possible to remove the redundancy while the latter allows learning lightweight and highly effective GCNs. These two constraints also act as regularizers that model structural relationships between network parameters in order to enhance both their generalization and lightweightness. Experiments conducted on the challenging task of skeleton-based hand-gesture recognition, shows the outperformance of the proposed lightweight GCNs against different baselines as well as the related work.

		H	L	L+orth	L+MP	L+orth+stc	L+MP
$K=2$	Accuracy (%)	84.17	83.30	84.52	84.52	83.65	81.56
	Pruning rate (%)	none	none	50	50	95	95
$K=4$	Accuracy (%)	82.95	83.82	85.21	83.13	85.73	82.95
	Pruning rate (%)	none	none	75	75	95	95
$K=8$	Accuracy (%)	72.69	83.82	85.04	83.65	**86.78**	84.00
	Pruning rate (%)	none	none	87	87	95	95

Table 1: Detailed performances, for different K, using handcrafted and learned connectivity w/o and with our constraints. We also compare these results with those of GCNs obtained using magnitude pruning (for the same pruning rates: $\lfloor(1-\frac{1}{K})\times 100\rfloor$ for L+orth vs. L+MP and $\lfloor(1-\frac{1}{n})\times 100\rfloor$ for L+orth+stc vs. L+MP), here H, L, orth, stc and MP stands respectively for handcrafted, learned, orthogonality, stochasticity and magnitude pruning.

Method	Color	Depth	Pose	Accuracy (%)
Two stream-color [31]	✓	✗	✗	61.56
Two stream-flow [31]	✓	✗	✗	69.91
Two stream-all [31]	✓	✗	✗	75.30
HOG2-depth [32]	✗	✓	✗	59.83
HOG2-depth+pose [32]	✗	✓	✓	66.78
HON4D [33]	✗	✓	✗	70.61
Novel View [34]	✗	✓	✗	69.21
1-layer LSTM [36]	✗	✗	✓	78.73
2-layer LSTM [36]	✗	✗	✓	80.14
Moving Pose [37]	✗	✗	✓	56.34
Lie Group [38]	✗	✗	✓	82.69
HBRNN [39]	✗	✗	✓	77.40
Gram Matrix [41]	✗	✗	✓	85.39
TF [42]	✗	✗	✓	80.69
JOULE-color [43]	✓	✗	✗	66.78
JOULE-depth [43]	✗	✓	✗	60.17
JOULE-pose [43]	✗	✗	✓	74.60
JOULE-all [43]	✓	✓	✓	78.78
Huang et al. [44]	✗	✗	✓	84.35
Huang et al. [45]	✗	✗	✓	77.57
Our best (table 1)	✗	✗	✓	**86.78**

Table 2: Comparison against state of the art methods.

5. REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton. "Deep learning." nature, pages 436–444, 2015.

[2] M. Jiu and H. Sahbi, "Semi supervised deep kernel design for image annotation," in *ICASSP*, 2015.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in NIPS, vol. 60, pages 1097–1105, 2012.

[4] C. Szegedy et al. "Going deeper with convolutions," in CVPR, 2015.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in CVPR, 2016.

[6] M. Jiu and H. Sahbi, “Laplacian deep kernel learning for image annotation,” in *ICASSP*, 2016.

[7] He et al. ”Mask r-cnn.” Proceedings of ICCV, 2017.

[8] G. Huang et al., “Densely connected convolutional networks,” in CVPR, 2017.

[9] M. Jiu and H. Sahbi, “Nonlinear deep kernel learning for image annotation,” in *IEEE TIP*, vol. 26(4), 2017.

[10] H. Sahbi, Coarse-to-fine deep kernel networks. In IEEE ICCV-W, 1131-1139, 2017.

[11] M. Jiu and H. Sahbi, “Deep representation design from deep kernel networks,” *Pattern Recognition*, vol. 88, pp. 447–457, 2019.

[12] Z. Zhang, P. Cui, and W. Zhu. ”Deep learning on graphs: A survey.” In IEEE TKDE, 2020.

[13] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun. Spectral networks and locally connected networks on graphs. arXiv preprint arXiv:1312.6203, 2013.

[14] M. Defferrard, X. Bresson, P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In NIPS, pages 3844–3852, 2016.

[15] T.N. Kipf, M. Welling. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907, 2016.

[16] Prates, Marcelo, et al. ”Learning to solve NP-complete problems: A graph neural network for decision TSP.” In AAAI, vol. 33, 2019.

[17] X. Dong, D. Thanou, M. Rabbat, and P. Frossard, Learning graphs from data: A signal representation perspective, arXiv preprint arXiv:1806.00848, 2018.

[18] S. Sardellitti et al., Graph topology inference based on sparsifying transform learning. IEEE TSP, 67(7), 2019.

[19] S.P. Chepuri, S. Liu, G. Leus, and A.O. Hero, Learning sparse graphs under smoothness prior, in ICASSP, 2017.

[20] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst, Learning Laplacian matrix in smooth graph signal representations, IEEE TSP, 64(23), pages 6160–6173, 2016.

[21] H. Sahbi, “Imageclef annotation with explicit context-aware kernel maps,” In *IJMIR*, pp. 113–128, 2015.

[22] B. Le Bars et al. Learning Laplacian Matrix from Bandlimited Graph Signals. In ICASSP, 2019

[23] L. Franceschi and M. Niepert and M. Pontil and X. He. Learning Discrete Structures for Graph Neural Networks. ICML, 2019.

[24] Yu Chen et al., Deep Iterative and Adaptive Learning for Graph Neural Networks, AAAI DLGMA, 2020.

[25] C. Li, Q. Zhong, D. Xie, and S. Pu. Co-occurrence feature learning from skeleton data for action recognition and detection with hierarchical aggregation. arXiv preprint arXiv:1804.06055, 2018.

[26] Z. Chenyi and Q. Ma. Dual graph convolutional networks for graph-based semi-supervised classification. Proceedings of WWW, 2018.

[27] Y. Li, R. Yu, C. Shahabi, and Y. Liu, Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In Proc. of ICLR, 2018.

[28] N. Yasunori. A note on Riemannian optimization methods on the Stiefel and the Grassmann manifolds. In NOLTA, vol 1, pages 349–352, 2005.

[29] G. Garcia-Hernando, S. Yuan, S. Baek, and T.-K. Kim. First Person Hand Action Benchmark with RGB-D Videos and 3D Hand Pose Annotations. In CVPR, 2018.

[30] L. Wang, H. Sahbi. Nonlinear Cross-View Sample Enrichment for Action Recognition. In *ECCV-W*, 2014.

[31] C. Feichtenhofer, A. P., and A. Zisserman. Convolutional Two-Stream Network Fusion for Video Action Recognition. In CVPR, 2016.

[32] E.Ohn-Barand, M.M.Trivedi. Hand Gesture Recognition in Real Time for Automotive Interfaces: A Multimodal Vision- Based Approach and Evaluations. IEEE TITS, 15(6), pages 2368–2377, 2014.

[33] O. Oreifej and Z. Liu. HON4D: Histogram of Oriented 4D Normals for Activity Recognition from Depth Sequences. In CVPR, pages 716–723, 2013.

[34] H. Rahmani and A. Mian. 3D Action Recognition from Novel Viewpoints. In CVPR, pages 1506–1515, 2016.

[35] L. Wang, H. Sahbi. Directed Acyclic Graph Kernels for Action Recognition. In *IEEE ICCV*, 2013.

[36] W. Zhu et al., Co-occurrence feature learning for skeleton based action recognition using regularized deep LSTM networks In AAAI, 2016.

[37] M. Zanfir et al., The Moving Pose: An Efficient 3D Kinematics Descriptor for Low-Latency Action Recognition and Detection. In ICCV, 2013.

[38] R. Vemulapalli, F. Arrate, and R. Chellappa. Human action recognition by representing 3D skeletons as points in a Lie group. In IEEE CVPR, 2014.

[39] Y. Du et al., Hierarchical recurrent neural network for skeleton based action recognition. In IEEE CVPR, 2015.

[40] L. Wang, H. Sahbi. Bags-of-Daglets for Action Recognition. In *IEEE ICIP*, 2014.

[41] X. Zhang et al., Efficient Temporal Sequence Comparison and Classification Using Gram Matrix Embeddings on a Riemannian Manifold. In CVPR, 2016.

[42] G. Garcia-Hernando and T.-K. Kim. Transition Forests: Learning Discriminative Temporal Transitions for Action Recognition. In CVPR, 2017.

[43] J. Hu et al., Jointly Learning Heterogeneous Features for RGB-D Activity Recognition. In CVPR, 2015

[44] Z. Huang and L. V. Gool. A Riemannian Network for SPD Matrix Learning. In AAAI, 2017

[45] Z. Huang, J. Wu, and L. V. Gool. Building Deep Networks on Grassmann Manifolds. In AAAI, 2018

[46] H. Sahbi. http://www-ia.lip6.fr/∼sahbi/proofSM.pdf. Supplementary material, 2021.