HAL
open science

# Learning Laplacians in Chebyshev Graph Convolutional Networks

Hichem Sahbi

## To cite this version:

Hichem Sahbi. Learning Laplacians in Chebyshev Graph Convolutional Networks. IEEE/CVF International Conference on Computer Vision (ICCV), DLGC, Oct 2021, Montreal (virtual), Canada. pp.2064-2075. hal-03431377

**HAL Id: hal-03431377**

**https://hal.science/hal-03431377v1**

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.

This ICCV workshop paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

# Learning Laplacians in Chebyshev Graph Convolutional Networks

Hichem Sahbi
Sorbonne University, CNRS, LIP6
F-75005, Paris, France
hichem.sahbi@sorbonne-universite.fr

## Abstract

*Spectral graph convolutional networks (GCNs) are particular deep models which aim at extending neural networks to arbitrary irregular domains. The principle of these networks consists in projecting graph signals using the eigen-decomposition of their Laplacians, then achieving filtering in the spectral domain prior to back-project the resulting filtered signals onto the input graph domain. However, the success of these operations is highly dependent on the relevance of the used Laplacians which are mostly handcrafted and this makes GCNs clearly sub-optimal. In this paper, we introduce a novel spectral GCN that learns not only the usual convolutional parameters but also the Laplacian operators. The latter are designed "end-to-end" as a part of a recursive Chebyshev decomposition with the particularity of conveying both the differential and the non-differential properties of the learned representations – with increasing order and discrimination power – without overparametrizing the trained GCNs. Extensive experiments, conducted on the challenging task of skeleton-based action recognition, show the generalization ability and the outperformance of our proposed Laplacian design w.r.t. different baselines (built upon handcrafted and other learned Laplacians) as well as the related work.*

## 1. Introduction

Deep learning is currently achieving a significant progress in computer vision and several related fields [38–41, 67–71]. The purpose of deep learning is to train multi-layered and highly nonlinear parametric models whose inputs correspond to vectorial data (images, etc.) and outputs to their classification or regression [44, 45]. One of the most popular models include convolutional neural networks (CNNs) which operate by shifting equivariant filters and measuring their responses across different image locations. While these operations are well defined on regular grids (namely images), their extension to irregular domains (such as skeletons in action recognition [20–22]) requires designing convolutions on general graph structures [46–49].

The difficulty in extending CNNs to irregular graphs stems from the eclectic properties of graphs and their heterogeneous topological properties which make convolutions *ill-posed*. A particular class of neural machines, known as graph convolutional networks (GCNs), has emerged and seeks to generalize convolutions to irregular graph structures [50]. Their principle consists in learning representations by aggregating signals through nodes and their neighbors, prior to apply convolutions on the resulting aggregates [27, 29, 50, 55, 64–66, 72]. Two categories of GCNs exist in the literature; spatial and spectral. In spatial GCNs [51–56], the representation of a given node is obtained by averaging the representations of its neighbors before applying convolutions using the inner product. In spectral GCNs [15, 28, 46, 47, 49, 57–61], convolutions proceed differently by first projecting filters and input graph signals using the eigen-decomposition of their Laplacians, prior to achieve filtering, and then back-projecting the resulting filtered signals onto the input graph domain [62, 63]. Whereas spectral GCNs make convolutions well defined compared to spatial ones, their drawback resides in the non-localized aspect of the learned filters and also the high complexity of Laplacian eigen-decomposition.

Other spectral GCNs have been introduced in the literature including Chebyshev networks [47] which consider instead localized convolutional filters using a recursive polynomial decomposition. The success of these GCNs is highly reliant on the choice of the Laplacian operators, and most of the existing ones are handcrafted or designed upon the inherent properties of the data. These properties correspond to preexisting (intrinsic) node-to-node relationships such as connectivity in 3D skeletons, links in social networks, protein connectivity in biological systems, etc. Nevertheless, handcrafted Laplacians are not sufficient in order to capture all node relationships as their design is oblivious to the tasks at hand. Indeed, in many real-world applications such as skeleton-based recognition, intrinsic relationships

are suitable to identify individuals using their anthropometric measurements while other extrinsic characteristics are more useful to recognize their dynamics and actions. Hence, extrinsic node-to-node relationships should be inferred in order to maximize the performances of the targeted applications (see Fig. 3). In other words, connectivity in Laplacians should be appropriately learned by including not only the available (intrinsic) node-to-node connections in graphs but also their inferred (extrinsic) relationships.

### 1.1. Related work

Laplacian — or equivalently graph — inference is generally ill-posed, NP-hard [1–3] and most of the existing approaches rely on constraints (including similarity, smoothness, sparsity, band-limitedness, etc.) for a better conditioning [4, 5, 73–86]. Particularly in GCNs, early methods [6,49,52] rely on predetermined node-to-node relationships using similarities or the inherent properties of the targeted applications in order to define Laplacian operators. However, in spite of being relatively effective, the potential of these operators is not fully explored as their design is either agnostic to the tasks at hand or achieved using the tedious cross validation. More recent advances aim at defining graph topology — and hence the underlying Laplacian — that best fits a given task [7–12, 15, 59]. For instance, the work in [11] 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 [12] using a weighted cosine similarity and edge thresholding.

Other solutions make improvement w.r.t. the original GCNs [49] by exploiting symmetric matrices [15] 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 [59] introduces a dual architecture with two parallel graph convolutional layers sharing the same parameters. This method considers a normalized adjacency matrix and a positive pointwise mutual information matrix to capture node co-occurrences through random walks sampled from graphs. In the particular context of skeleton-based action recognition, GCNs have been increasingly used [13, 14, 24–26, 35, 36, 42, 43] as they explicitly model, with a better interpretability, the spatial and temporal connectivity among joints. However, while all skeleton-joints contribute in motion, only a few of them are actually relevant to recognize the targeted action categories; hence, other work focuses on learning more complete spatial and temporal joint co-occurrences for skeleton data [15, 30, 34, 37].

### 1.2. Contribution

In this paper, we introduce a novel framework that learns graph topology and Laplacians as a part of GCN and Chebyshev basis design. This basis is expressed using an efficient recursive form evaluated on a single shared Laplacian which conveys both the differential and non-differential properties of the learned graph representations. This Chebyshev basis also captures the statistical properties of the learned representations, with increasing order and discrimination power, without increasing the actual number of training parameters in the resulting GCNs. Different settings of our design are considered including symmetry and orthogonality that further constrain the learned Laplacians and enhance the generalization capacity of our GCNs. Experiments, conduced on the challenging task of skeleton-based human action and hand gesture recognition, show the high accuracy and the outperformance of our method w.r.t. different baselines as well as the related work.

Our proposed method in this paper is different, from all the aforementioned related work, at least in two aspects; on the one hand, none of this related work considers Laplacian learning as a part of Chebyshev basis design. On the other hand, existing methods consider multiple independent matrix operators that capture the actual topology of the input graphs and increase the discrimination power of the learned GCN representations, but this comes at the expense of overparametrized networks and the risk of overfitting. In contrast, our Chebyshev basis design increases the discrimination power of the representations (that capture different hops in graphs), without overparametrizing the trained networks, as the learned Laplacian parameters are shared through all the Chebyshev polynomials. Besides, making the Chebyshev basis[1] orthogonal acts as a regularizer that controls the actual number of training parameters and enhances further the generalization power of our GCNs as corroborated later in experiments.

## 2. Chebyshev Convolutional Networks

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $|\mathcal{V}| = n$, $|\mathcal{E}|$ being respectively the number of its vertices and edges, and $\mathbf{L}$ the Laplacian of $\mathcal{G}$. For instance, $\mathbf{L}$ could be the random walk Laplacian defined as $\mathbf{L} = \mathbf{I}_n - \mathbf{A}[\mathbf{D}^{-1}(\mathbf{A})]$ where (i) $\mathbf{I}_n$ is an $n \times n$ identity matrix, (ii) $\mathbf{A}$ an adjacency matrix with each entry $\mathbf{A}_{uu'} > 0$ iff $(u, u') \in \mathcal{E}$ and $0$ otherwise, and (iii) $\mathbf{D}(\mathbf{A})$ a diagonal degree matrix with each entry $[\mathbf{D}(\mathbf{A})]_{uu} = \sum_v \mathbf{A}_{vu}$. Let $\mathbf{U}\Lambda\mathbf{U}^\top$ be the eigen-decomposition of $\mathbf{L}$, with $\mathbf{U}$, $\Lambda$ being respectively the matrix of eigenvectors (graph Fourier basis) and the

[1]The generative aspect of our basis makes it possible to capture different hops of neighbors without increasing the actual number of training parameters and this enhances the discrimination power of the learned representation as shown through this paper.

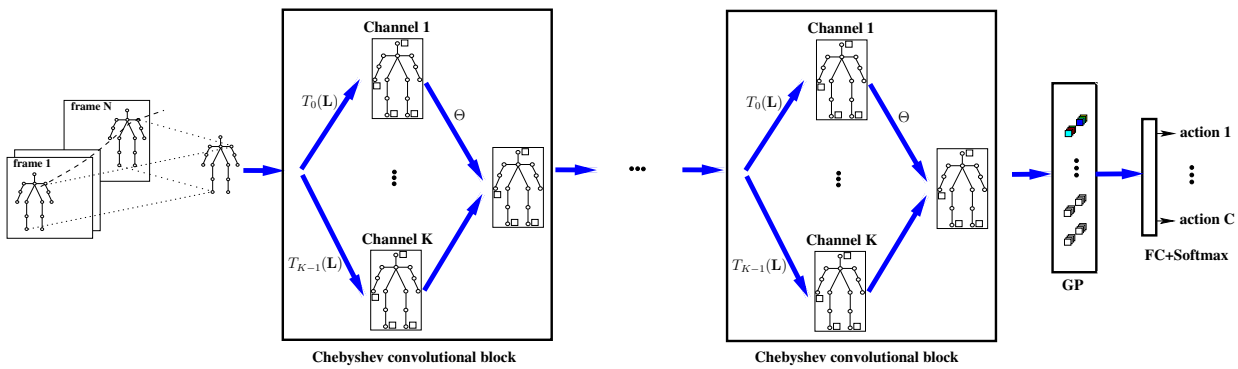


Figure 1. This figure shows the architecture of our Chebyshev Convolutional Network. In each convolutional block, the Chebyshev basis $\{T_k(.)\}_k$ is first evaluated on the Laplacian $\mathbf{L}$, then multiplied by the input graph signal $\psi(\mathcal{V})$, and finally aggregated using the parameters $\theta$. These Chebyshev convolutional blocks are followed by global average pooling prior to softmax classification. Note that the Laplacian $\mathbf{L}$ is shared across the basis $\{T_k(.)\}_k$ and through the Chebyshev convolutional blocks; in practice, only a very few convolutional blocks are necessary as the basis $\{T_k(.)\}_k$ already captures different hops/depths in graphs. **(Better to zoom the pdf)**.

diagonal matrix of its eigenvalues; spectral graph convolution is a well defined operator (see for instance [50]) which is achieved by first projecting a given graph signal $\psi(.)$ using the eigen-decomposition of $\mathbf{L}$, and then multiplying the resulting projection by a convolutional filter prior to back-project the result in the original signal space.

Formally, the convolutional operator $\star_{\mathcal{G}}$ (rewritten for short as $\star$) on a given graph signal $\psi(\mathcal{V}) \in \mathbb{R}^{s\times n}$ is $(\psi \star g_\theta)_{\mathcal{V}} = \mathbf{U}\ g_\theta(\Lambda)\mathbf{U}^\top \psi(\mathcal{V})^\top$; here $^\top$ is the matrix transpose operator and $g_\theta$ denotes a non-parametric convolutional filter defined as $g_\theta(\Lambda) = diag(\theta)$ with $\theta \in \mathbb{R}^n$. As this filter is not localized, we consider instead [47]

$$(\psi \star g_\theta)_{\mathcal{V}} = \sum_{k=0}^{K-1} \theta_k\, T_k(\mathbf{L})\ \psi(\mathcal{V})^\top, \tag{1}$$

with $\theta = (\theta_1 \dots \theta_K)^\top \in \mathbb{R}^K$ being the learned convolutional filter parameters and $T_k$ the $k$-th order Chebyshev polynomial recursively defined as $T_k(\mathbf{L}) = 2\mathbf{L}\circ T_{k-1}(\mathbf{L}) - T_{k-2}(\mathbf{L})$, with $T_k(\mathbf{L}) \in \mathbb{R}^{n\times n}$, $T_0(\mathbf{L}) = \mathbf{I}_n$, $T_1(\mathbf{L}) = \mathbf{L}$ and $\circ$ the Hadamard (element-wise) matrix product. When $\mathbf{L}$ is the combinatorial Laplacian, we consider in practice a rescaled version as $2\mathbf{L}/\lambda_{\max} - \mathbf{I}_n$ (instead of $\mathbf{L}$ with $\lambda_{\max}$ being the largest eigenvalue of $\mathbf{L}$) in order to guarantee the orthogonality of the basis $\{T_k(\mathbf{L})\}_k$; see again [47] and later (in section 3.2) the general rescaling of any Laplacian that guarantees orthogonality of the Chebyshev basis. The whole architecture of this convolution is described in Fig. 1.

## 3. Our Laplacian Design

Considering the tensor of the Chebyshev polynomials $\{T_k(\mathbf{L})\}_k$ and the loss $\mathcal{L}$ associated to a given classification task, we turn the design of the Laplacian $\mathbf{L}$ (thereby the Chebychev basis $\{T_k(\mathbf{L})\}_k$) as a part of GCN training. Considering the gradient of $\mathcal{L}$ w.r.t. the Chebyshev terms, denoted as $\nabla_k\mathcal{L} = \frac{\partial \mathcal{L}}{\partial T_k(\mathbf{L})}$, and since $\mathbf{L}$ is shared across $\{T_k(\mathbf{L})\}_k$, one may write

$$\frac{\partial \mathcal{L}}{\partial \mathbf{L}} = \mathbf{vec}^{-1}\Big(\sum_{k=0}^{K-1} \mathbf{J}_k\ \mathbf{vec}(\nabla_k \mathcal{L})\Big), \tag{2}$$

being $\mathbf{J}_k \in \mathbb{R}^{n^2\times n^2}$ the diagonal Jacobian matrix whose entry $[\mathbf{J}_k]_{ij,ij} = \frac{\partial [T_k(\mathbf{L})]_{ij}}{\partial \mathbf{L}_{ij}}$ and $\mathbf{vec}(.)$ a vectorization that appends the entries of a given matrix using the x-y order in $\mathbf{J}_k$, and $\mathbf{vec}^{-1}$ its inverse. In the above equation, one may show that each entry of the Jacobian $\mathbf{J}_k$ can be recursively obtained as

$$\begin{cases} 0 & k=0 \\ 1 & k=1 \\ 2\big[[T_{k-1}(\mathbf{L})]_{ij} + \mathbf{L}_{ij}\frac{\partial [T_{k-1}(\mathbf{L})]_{ij}}{\partial \mathbf{L}_{ij}}\big] - \frac{\partial [T_{k-2}(\mathbf{L})]_{ij}}{\partial \mathbf{L}_{ij}} & k\geq 2, \end{cases} \tag{3}$$

so $\mathbf{L}$ can be updated using Eqs (2), (3) and gradient descent. However, as designed above, the learned matrix $\mathbf{L}$ is not guaranteed to be a valid Laplacian. In what follows, we consider reparametrization which constrains $\mathbf{L}$ to be valid.

### 3.1. Laplacian Reparametrization

We turn the design of $\mathbf{L}$ into the learning of the adjacency matrix $\mathbf{A}$ while guaranteeing the resulting matrix $\mathbf{L}$ to be a valid Laplacian. Different reparameterizations are considered including $\mathbf{L} = \mathbf{D}(\mathbf{A}) - \mathbf{A}$ which corresponds to the combinatorial form. If we constrain $\mathbf{A}$ to be column-stochastic, then $\mathbf{L}$ corresponds to the random walk graph Laplacian. We also consider other reparametrizations including the normalized defined as $\mathbf{L} = \mathbf{I}_n - [\mathbf{D}(\mathbf{A}^\top)]^{-\frac{1}{2}}\mathbf{A}[\mathbf{D}(\mathbf{A})]^{-\frac{1}{2}}$ (see table 1).

| Constraints | Reparametrization | Jacobian |
|---|---|---|
| COMB | $\mathbf{D}(\mathbf{A}^\top) - \mathbf{A}$ | $[\mathbf{J}_{\text{c}}]_{ij,pq} = 1_{\{i=j,p\neq q\}} - 1_{\{i\neq j\}}$ |
| NDRW | $\mathbf{A}\,[\mathbf{D}(\mathbf{A})]^{-1}$ | $[\mathbf{J}_{\text{ndrw}}]_{ij,pq} = 1_{\{j=q\}}\,(\delta_{ip} - \mathbf{L}_{ij})\,[\mathbb{1}_n\,\mathbf{D}(\mathbf{A})^{-1}]_{pq}$ |
| DRW | $\mathbf{I}_n - \mathbf{A}\,[\mathbf{D}(\mathbf{A})]^{-1}$ | $[\mathbf{J}_{\text{drw}}]_{ij,pq} = 1_{\{j=q\}}\,(\mathbf{L}_{ij} - \delta_{ip})\,[\mathbb{1}_n\,\mathbf{D}(\mathbf{A})^{-1}]_{pq}$ |
| NDN | $[\mathbf{D}(\mathbf{A}^\top)]^{-\frac{1}{2}}\mathbf{A}\,[\mathbf{D}(\mathbf{A})]^{-\frac{1}{2}}$ | $[\mathbf{J}_{\text{ndn}}]_{ij,pq} = 1_{\{i=p\vee j=q\}}\,\frac{\mathbf{L}_{ij}}{2\mathbf{A}_{pq}}\,\big(2\delta_{ip}\delta_{jq} - [\mathbf{D}(\mathbf{A}^\top)^{-1}\mathbf{A} + \mathbf{A}\,\mathbf{D}(\mathbf{A})^{-1}]_{pq}\big)$ |
| DN | $\mathbf{I}_n - [\mathbf{D}(\mathbf{A}^\top)]^{-\frac{1}{2}}\mathbf{A}\,[\mathbf{D}(\mathbf{A})]^{-\frac{1}{2}}$ | $[\mathbf{J}_{\text{dn}}]_{ij,pq} = 1_{\{i=p\vee j=q\}}\,\frac{\mathbf{L}_{ij}}{2\mathbf{A}_{pq}}\,\big([\mathbf{D}(\mathbf{A}^\top)^{-1}\mathbf{A} + \mathbf{A}\,\mathbf{D}(\mathbf{A})^{-1}]_{pq} - 2\delta_{ip}\delta_{jq}\big)$ |
| Symmetry | $\mathbf{A} + \mathbf{A}^\top$ | $[\mathbf{J}_{\text{s}}]_{ij,pq} = 1_{\{(i=p,j=q)\vee(i=q,j=p)\}}$ |
| S-COMB | $\mathbf{D}(\mathbf{A}+\mathbf{A}^\top) - (\mathbf{A}+\mathbf{A}^\top)$ | $\mathbf{J}_{\text{sc}} = \mathbf{J}_{\text{c}}\,\mathbf{J}_{\text{s}}$ |
| S-NDRW | $(\mathbf{A}+\mathbf{A}')\,[\mathbf{D}(\mathbf{A}+\mathbf{A}^\top)]^{-1}$ | $\mathbf{J}_{\text{sndrw}} = \mathbf{J}_{\text{ndrw}}\,\mathbf{J}_{\text{s}}$ |
| S-DRW | $\mathbf{I}_n - (\mathbf{A}+\mathbf{A}^\top)\,[\mathbf{D}(\mathbf{A}+\mathbf{A}^\top)]^{-1}$ | $\mathbf{J}_{\text{sdrw}} = \mathbf{J}_{\text{drw}}\,\mathbf{J}_{\text{s}}$ |
| S-NDN | $[\mathbf{D}(\mathbf{A}+\mathbf{A}^\top)]^{-\frac{1}{2}}(\mathbf{A}+\mathbf{A}^\top)\,[\mathbf{D}(\mathbf{A}+\mathbf{A}^\top)]^{-\frac{1}{2}}$ | $\mathbf{J}_{\text{sndn}} = \mathbf{J}_{\text{ndn}}\,\mathbf{J}_{\text{s}}$ |
| S-DN | $\mathbf{I}_n - [\mathbf{D}(\mathbf{A}+\mathbf{A}^\top)]^{-\frac{1}{2}}(\mathbf{A}+\mathbf{A}^\top)\,[\mathbf{D}(\mathbf{A}+\mathbf{A}^\top)]^{-\frac{1}{2}}$ | $\mathbf{J}_{\text{sdn}} = \mathbf{J}_{\text{dn}}\,\mathbf{J}_{\text{s}}$ |

Table 1. Different reparametrizations and the underlying Jacobians. In this table, COMB stands for "Combinatorial" Laplacian, NDRW for "Non Differential Random Walk", DRW for "Differential Random Walk", NDN for "Non Differential Normalized" Laplacian, and DN for "Differential Normalized" one. The symmetric variants of these Laplacians are prefixed by "S".

For any reparametrization of $\mathbf{L}$, the chain rule leads to

$$\frac{\partial \mathcal{L}}{\partial \mathbf{A}} = \mathbf{vec}^{-1}\left(\mathbf{J}\ \mathbf{vec}\left(\frac{\partial \mathcal{L}}{\partial \mathbf{L}}\right)\right), \tag{4}$$

with $\frac{\partial \mathcal{L}}{\partial \mathbf{L}}$ obtained from Eq. 2 and $\mathbf{J}$ being a sparse Jacobian matrix whose entry $[\mathbf{J}]_{ij,pq} = [\frac{\partial \mathbf{L}_{ij}}{\partial \mathbf{A}_{pq}}]_{ij,pq}$; this matrix is given in table 1 for different Laplacian settings including the combinatorial and random walk which respectively capture the differential and non-differential properties of node features. We also consider the differential random walk – as a combination of these two Laplacians – obtained by plugging the latter into the former. All these Laplacians are built upon either symmetric or non-symmetric matrices $\mathbf{A}$. Note that symmetry is obtained using weight sharing, i.e., by constraining the upper and the lower triangular parts of $\mathbf{A}$ to share the same entries. This is guaranteed by considering a reparametrization as $\mathbf{A} + \mathbf{A}^\top$ (with $\mathbf{A}$ being now a free matrix) and by tying pairwise symmetric entries of the gradient $\frac{\partial \mathcal{L}}{\partial \mathbf{A}}$; this is equivalently obtained by multiplying the original gradient $\frac{\partial \mathcal{L}}{\partial \mathbf{A}}$ by the Jacobian $[\mathbf{J}_{\text{s}}]_{ij,pq} = 1_{\{(i=p,j=q)\vee(i=q,j=p)\}}$ which is again extremely sparse and highly efficient to evaluate.

### 3.2. Orthogonality

Learning multiple matrices $\{T_k(\mathbf{L})\}_{k=0}^{K-1}$ allows us to capture different graph topologies when achieving aggregation and convolution, and this enhances the discrimination power of the GCN representations without increasing the actual number of training parameters (as also shown later in experiments). However, if aggregation produces, for a given $u \in \mathcal{V}$, linearly dependent vectors $\mathcal{X}_u = \{\sum_{u'}[T_k(\mathbf{L})]_{uu'}\ \psi(u')\}_k$, then convolution will also generate linearly dependent representations with an overestimated number of training parameters in the null space of $\mathcal{X}_u$. Besides, matrices $\{T_1(\mathbf{L}), \ldots, T_K(\mathbf{L})\}$ used for aggregation, may also correspond to overlapping and redundant neighborhoods.

Provided that $\{\psi(u')\}_{u'\in\mathcal{N}_r(u)}$ are linearly independent, and $K$ upper-bounded by $\mathbf{rank}(\{\psi(u')\}_{u'\in\mathcal{N}_r(u)}) \leq \min(|\mathcal{V}|, s)$, the condition that makes vectors in $\mathcal{X}_u$ linearly independent reduces to orthogonality, i.e., $\langle T_k(\mathbf{L}), T_{k'}(\mathbf{L})\rangle_F = 0$, $\forall k \neq k'$, with $\langle .,. \rangle_F$ being the Hilbert-Schmidt (or Frobenius) inner product defined as $\langle T_k(\mathbf{L}), T_{k'}(\mathbf{L})\rangle_F = \mathbf{tr}(T_k(\mathbf{L})^\top T_{k'}(\mathbf{L}))$ with $\mathbf{tr}(.)$ being the matrix trace operator. A sufficient condition that guarantees the orthogonality of the Chebyshev basis consists in taking the Laplacian $2(\mathbf{L} - \lambda_{\text{min}}\mathbf{I}_n)/(\lambda_{\text{max}} - \lambda_{\text{min}}) - \mathbf{I}_n$ instead of $\mathbf{L}$ with $\lambda_{\text{min}}$ (resp. $\lambda_{\text{max}}$) being the smallest (resp. largest) eigenvalue of $\mathbf{L}$, and this guarantees the eigenvalues of the resulting matrix to be in $[-1, +1]$ and also the minimality of $\{T_k(\mathbf{L})\}_k$ (see for instance [47]). It is easy to see that this normalization equates the rescaled Laplacian shown in section 2, i.e., on the combinatorial setting, as its smallest eigenvalue is zero.

## 4. Experiments

In this section, we evaluate the performance of our GCN network for the task of action recognition [93, 99] using two challenging skeleton datasets; SBU Interaction [88] and First-Person Hand Action (FPHA) [87]. The purpose is to show the relevance of our Laplacian design and its comparison against different handcrafted Laplacians and learned ones as well as more general related work in action recognition.

### 4.1. Datasets and implementation details

**Dataset description.** SBU is an interaction dataset acquired (under relatively well controlled conditions) using the Microsoft Kinect sensor; it includes in total 282 moving skeleton sequences (performed by two interacting individuals) belonging to 8 categories: "approaching", "departing", "pushing", "kicking", "punching", "exchanging objects",

"hugging", and "hand shaking". Each pair of interacting individuals corresponds to two 15 joint skeletons and each joint is encoded with a sequence of its 3D coordinates across video frames. In this dataset, we consider the same evaluation protocol as the one suggested in the original dataset release [88] (i.e., train-test split).
The FPHA dataset includes 1175 skeletons belonging to 45 action categories which are performed by 6 different individuals in 3 scenarios. In contrast to SBU, action categories are highly variable with inter and intra subject variability including style, speed, scale and viewpoint. Each skeleton includes 21 hand joints and each joint is again encoded with a sequence of its 3D coordinates across video frames. We evaluate the performance of our method using the 1:1 setting proposed in [87] with 600 action sequences for training and 575 for testing. In all these experiments, we report the average accuracy over all the classes of actions.

**Skeleton normalization.** Let $S^t = \{p_1^t, \ldots, p_n^t\}$ denote the 3D skeleton coordinates at frame $t$. Without a loss of generality, we consider a particular order so that $p_1^t$, $p_2^t$ and $p_3^t$ correspond to three reference joints (e.g., neck, left shoulder and right shoulder for SBU dataset); as shown in Fig. 3, this corresponds to joints 2, 4 and 7 for SBU and 1, 3 and 5 for FPHA. As the relative distance between these 3 joints is stable w.r.t. any motion, these 3 joints are used in order to estimate the rigid motion (similarity transformation) for skeleton normalization (see also [89]). Each graph sequence is processed in order to normalize its 3D coordinates using a similarity transformation; the translation parameters $\mathbf{t} = (t_x, t_y, t_z)$ of this transformation correspond to the shift that makes the reference point $(p_2^0 + p_3^0)/2$ coincide with the origin while the rotation parameters $(\theta_x, \theta_y, \theta_z)$ are chosen in order to make the plane formed by $p_1^0$, $p_2^0$ and $p_3^0$ coplanar with the x-y plane and the vector $p_2^0 - p_3^0$ colinear with the x-axis. Finally, the scaling $\gamma$ of this similarity is chosen to make the $\|p_2^0 - p_3^0\|_2$ constant through all the action instances. Hence, each normalized joint is transformed as $\hat{p}_i^t = \gamma(p_i^t - \mathbf{t})R_x(\theta_x)R_y(\theta_y)R_z(\theta_z)$ with $R_x$, $R_y$, $R_z$ being rotation matrices along the $x$, $y$ and $z$ axes respectively.

**Input graphs.** Considering a sequence of normalized skeletons $\{S^t\}_t$, each joint sequence $\{\hat{p}_j^t\}_t$ in these skeletons defines a labeled trajectory through successive frames (see Fig. 2). Given a finite collection of trajectories, we consider the input graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where each node $v_j \in \mathcal{V}$ corresponds to the labeled trajectory $\{\hat{p}_j^t\}_t$ and an edge $(v_j, v_i) \in \mathcal{E}$ exists between two nodes iff the underlying trajectories are spatially neighbors. Each trajectory (i.e., node in $\mathcal{G}$) is processed using *temporal chunking*: first, the total duration of a sequence (video)

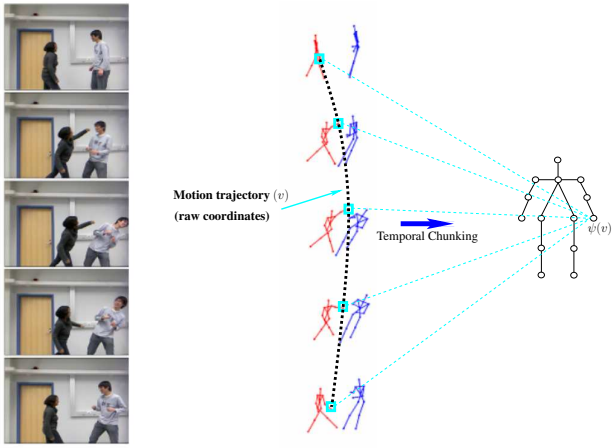


Figure 2. This figure shows the whole keypoint tracking and description process.

is split into $M$ equally-sized temporal chunks ($M = 4$ in practice), then the normalized joint coordinates $\{\hat{p}_j^t\}_t$ of the trajectory $v_j$ are assigned to the $M$ chunks (depending on their time stamps) prior to concatenate the averages of these chunks; this produces the description of $v_j$ (again denoted as $\psi(v_j) \in \mathbb{R}^s$ with $s = 3 \times M$) and $\{\psi(v_j)\}_j$ constitutes the raw description of nodes in a given sequence. Note that two trajectories $v_j$ and $v_i$, with similar joint coordinates but arranged differently in time, will be considered as very different when using temporal chunking. Note also that beside being compact and discriminant, this temporal chunking gathers advantages – while discarding drawbacks – of two widely used families of techniques mainly *global averaging techniques* (invariant but less discriminant) and *frame resampling techniques* (discriminant but less invariant). Put differently, temporal chunking produces discriminant raw descriptions that preserve the temporal structure of trajectories while being *frame-rate* and *duration* agnostic.

**Implementation settings.** We trained the GCN networks end-to-end using the Adam optimizer [101] for 1,800 epochs with a batch size equal to 200 for SBU and 600 for FPHA, a momentum of 0.9 and a global learning rate (denoted as $\nu(t)$) inversely proportional to the speed of change of the 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 $\nu(t) \leftarrow \nu(t-1)/0.99$). All these experiments are run on a GeForce GTX 1070 GPU device (with 8 GB memory) and neither dropout nor data augmentation are used.

## 4.2. Baselines

We compare the performances of our GCN w.r.t. different Chebyshev basis settings which are defined upon handcrafted and learned Laplacians, as well as against totally learned Laplacian basis. Note that all these Laplacians are combined with symmetry and orthogonality settings as described earlier.

**Handcrafted Laplacians (HL).** All the Chebyshev terms $\{T_k(.)\}_k$ are evaluated upon a *handcrafted* Laplacian $\mathbf{L}$ which in turns depends on a fixed adjacency matrix $\mathbf{A}$ (set using the original input graph).

**Multi-Laplacians (ML).** In this configuration, the Laplacian used in the Chebyshev terms is trained as a weighted *combination* of the handcrafted variants of the Laplacians in table 1 (built upon the fixed matrix $\mathbf{A}$). Note that orthogonality is obtained by normalizing the final learned Laplacian while symmetry is enforced in the handcrafted adjacency matrix $\mathbf{A}$.

**Totally Learned Laplacians (TLL).** In this variant, $K$ independent Laplacians $\{\mathbf{L}_k\}_k$ (and hence the underlying adjacency matrices) are learned. In contrast to the handcrafted setting, orthogonality and symmetry are obtained as a part of the optimization process (as already discussed in sections 3.1 and 3.2).

| Laplacians Settings | | Differential COMB | Non-Differential NDRW | Non-Differential NDN | Combined DRW | Combined DN |
|---|---|---|---|---|---|---|
| $K=2$ | HL | 96.9231 | 96.9231 | 96.9230 | 93.8462 | 96.9230 |
| | TLL | 96.9231 | 95.3846 | 98.4615 | 96.9231 | 98.4615 |
| | Our | 98.4615 | 98.4615 | 96.9230 | 96.9231 | 98.4615 |
| $K=4$ | HL | 95.3846 | 93.8462 | 96.9231 | 96.9230 | 96.9230 |
| | TLL | 96.9231 | 95.3846 | 98.4615 | 98.4615 | 96.9230 |
| | Our | 98.4615 | **100.000** | 98.4615 | 98.4615 | 98.4615 |
| $K=8$ | HL | 96.9231 | 98.4615 | 96.9231 | 96.9230 | 96.9230 |
| | TLL | 96.9231 | 98.4615 | 98.4615 | 93.8462 | 96.9230 |
| | Our | 96.9231 | 98.4615 | 98.4615 | 98.4615 | 98.4615 |

Table 2. Detailed performances on SBU using Chebyshev networks with handcrafted (HL) and learned Laplacians (Our), and using totally learned Laplacians (TLL). These performances are shown for $K \in \{2, 4, 8\}$ and for different reparametrizations of the Laplacians including differential (COMB), and non differential (NDRW, NDN) as well as their combinations (DRW, DN); see again Table. 1. Note that both symmetry and orthogonality constraints are used in these results.

## 4.3. Ablation study and comparison

Tables 2, 3 show a comparison of our GCN-based action recognition against the aforementioned GCN baselines, i.e. based on Handcrafted Laplacians and Totally Learned ones (performances with Multi-Laplacians are rather shown in Table 5); these comparisons are shown for different $K \in \{2, 4, 8\}$. From all these results, we observe a clear gain of our Chebyshev-based Laplacian design w.r.t. these baselines; at least one of the setting (namely $K = 4$) provides a significant gain. Table 4 shows an ablation study, where the impact of each component of our GCN (Laplacians, symmetry and orthogonality) is observed separately and jointly. From these results, we observe a positive impact when constraining the learned matrices to be symmetric and orthogonal; this gain is noticeable with non-differential Laplacians on SBU and with combined (differential/non-differential) ones on FPHA and this clearly shows the complementary aspect of these two Laplacian settings mainly on challenging datasets (i.e., FPHA). Again, this gain reaches the highest values when $K$ is sufficiently (not very) large and this follows the small size of the original skeletons (diameter and dimensionality of the graphs and the signal) used for action recognition which constrains the required number of Laplacian terms in the Chebyshev decomposition. Hence, with few Chebyshev terms, our method is able to learn relevant Laplacians and representations for action recognition.

| Laplacians Settings | | Differential COMB | Non-Differential NDRW | Non-Differential NDN | Combined DRW | Combined DN |
|---|---|---|---|---|---|---|
| $K=2$ | HL | 85.9130 | 85.3913 | 85.5652 | 85.3913 | 84.8695 |
| | TLL | 85.5652 | 86.4348 | 85.7391 | 85.3913 | 86.0869 |
| | Our | 85.3913 | 85.7391 | 85.5652 | 85.5652 | 85.7391 |
| $K=4$ | HL | 86.4348 | 84.1739 | 85.9130 | 84.0000 | 84.5217 |
| | TLL | 84.3478 | 85.3913 | 86.4347 | 85.0435 | 85.5652 |
| | Our | 85.2174 | 85.3913 | 85.7391 | **87.1304** | **87.3043** |
| $K=8$ | HL | 85.2174 | 83.8261 | 86.0869 | 84.6957 | 85.7391 |
| | TLL | 84.5217 | 85.5652 | 85.7391 | 85.0435 | 84.8695 |
| | Our | 84.6957 | 86.9565 | 86.7826 | 84.6957 | 84.5217 |

Table 3. Same caption as Table. 2 on the FPHA database.

| Dataset | Constraints Sym | Constraints Orth | Differential COMB | Non-Differential NDRW | Non-Differential DRW | Combined NDN | Combined DN | Avg. perf. |
|---|---|---|---|---|---|---|---|---|
| SBU | ✗ | ✗ | 90.76 | 98.46 | 98.46 | 96.92 | 98.46 | 96.61 |
| | ✓ | ✗ | 95.38 | 100.0 | 98.46 | 98.46 | 98.46 | 98.15 |
| | ✗ | ✓ | 95.38 | 96.92 | 98.46 | 96.92 | 98.46 | 97.23 |
| | ✓ | ✓ | 98.46 | **100.00** | 98.46 | 98.46 | 98.46 | **98.76** |
| Avg. | - | - | 95.00 | **98.84** | 98.46 | 97.69 | 98.46 | - |
| FPHA | ✗ | ✗ | 79.30 | 85.21 | 86.60 | 86.78 | 86.43 | 84.86 |
| | ✓ | ✗ | 84.00 | 85.73 | 85.73 | 86.43 | 86.08 | 85.60 |
| | ✗ | ✓ | 83.13 | 85.39 | 85.73 | 86.78 | 86.78 | 85.56 |
| | ✓ | ✓ | 85.21 | 85.39 | 85.73 | **87.13** | **87.30** | **86.15** |
| Avg. | - | - | 82.91 | 85.43 | 85.95 | **86.78** | **86.65** | - |

Table 4. Ablation study on SBU and FPHA databases, when symmetry (sym) and orthogonality (orth) are taken separately and when combined (in these results $K = 4$).

Our proposed Laplacian design avoids the strong bias about the handcrafted adjacency matrices which are rather suitable to capture the anthropometric characteristics of skeletons and less optimal for action recognition. On another hand, Tables. 2, 3 and 4 show that our Laplacian design makes it possible to capture better the topology of the graph data (i.e., the neighborhood system defined by the learned Laplacian and its underlying adjacency matrix

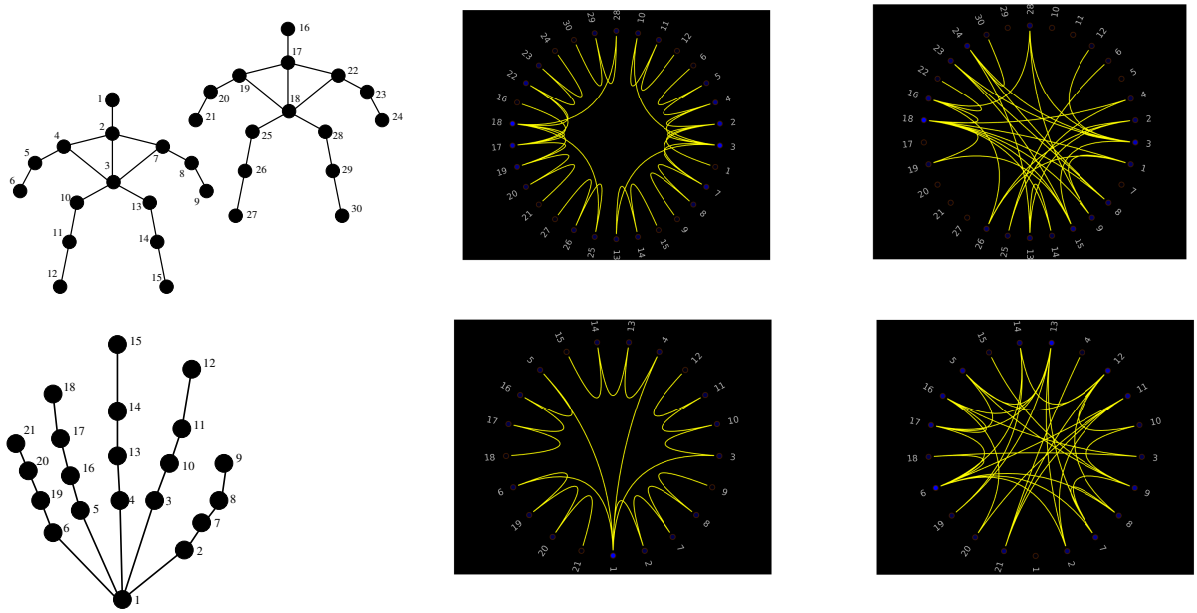

Figure 3. This figure shows original skeletons (left) with their intrinsic node-to-node relationships useful for *individual identification* (middle), and an example of the adjacency matrix associated to the learned Laplacian which shows the extrinsic node-to-node relationships found to be the most discriminating for *skeleton-based action recognition* when using our proposed method (the exact setting corresponds to Tables 2 and 3, using NDRW Laplacian with $K = 4$). **(Better to zoom the PDF version to view the learned node-to-node relationships).**

**A**). In contrast, the baselines are limited when connectivity is handcrafted and also when learned using totally trained Laplacians, as this results either into a biased Laplacian or into a larger number of training parameters, while Chebyshev provides a compromise between these two extreme cases: indeed, it enhances the discrimination power of the representation without increasing the actual number of training parameters. In sum, the gain of our GCN results from (i) the relative flexibility of the proposed design which allows learning complementary aspects of graph topology (through the Chebyshev basis), and also (ii) the regularization effect of our constraints (Laplacian weight sharing in Chebyshev, Laplacian reparametrization, orthogonality and symmetry) which all mitigate overfitting.

Finally, we compare the classification performances of our GCN against other, and more general, related methods in action recognition ranging from sequence based such as LSTM and GRU [18, 90, 104] to deep graph (non-vectorial) methods [24], etc. (see related work in tables 5 and 6). From the results in these tables, our GCN brings a noticeable gain w.r.t. related state of the art methods.

| Method | Accuracy (%) |
|---|---|
| Raw Position [88] | 49.7 |
| Joint feature [32] | 86.9 |
| CHARM [33] | 86.9 |
| H-RNN [16] | 80.4 |
| ST-LSTM [17] | 88.6 |
| Co-occurrence-LSTM [30] | 90.4 |
| STA-LSTM [18] | 91.5 |
| ST-LSTM + Trust Gate [17] | 93.3 |
| VA-LSTM [19] | 97.6 |
| GCA-LSTM [104] | 94.9 |
| Riemannian manifold. traj [103] | 93.7 |
| DeepGRU [90] | 95.7 |
| RHCN + ACSC + STUFE [24] | 98.7 |
| Multi-Laplacians (ML baseline [61]) | 98.4 |
| Our best (table 2) | **100** |

Table 5. Comparison against state of the art methods using the SBU database.

## 5. Conclusion

We introduce in this paper a novel Chebyshev-based Laplacian design for graph convolutional networks (GCNs). The learned Laplacians capture the most influencing interactions between body parts in skeleton action recognition. The strength of our method resides in its ability to learn shared Laplacians which are embedded in a Chebyshev basis that increases the discrimination power of our graph representation. In contrast to usual deep networks, the parameters of our GCNs are interpretable as their design is constrained "by construction" and this also brings a regularization effect that mitigates overfitting. Indeed, our Laplacian weight sharing and reparametrization — together with symmetry and orthogonality constraints — enhance the representational power of our learned graph features without increasing the actual number of training parameters in the resulting GCNs. Several operators are considered in-

| Method | Color | Depth | Pose | Accuracy (%) |
|---|---|---|---|---|
| Two stream-color [91] | ✓ | ✗ | ✗ | 61.56 |
| Two stream-flow [91] | ✓ | ✗ | ✗ | 69.91 |
| Two stream-all [91] | ✓ | ✗ | ✗ | 75.30 |
| HOG2-depth [92] | ✗ | ✓ | ✗ | 59.83 |
| HOG2-depth+pose [92] | ✗ | ✓ | ✓ | 66.78 |
| HON4D [94] | ✗ | ✓ | ✗ | 70.61 |
| Novel View [95] | ✗ | ✓ | ✗ | 69.21 |
| 1-layer LSTM [30] | ✗ | ✗ | ✓ | 78.73 |
| 2-layer LSTM [30] | ✗ | ✗ | ✓ | 80.14 |
| Moving Pose [96] | ✗ | ✗ | ✓ | 56.34 |
| Lie Group [23] | ✗ | ✗ | ✓ | 82.69 |
| HBRNN [16] | ✗ | ✗ | ✓ | 77.40 |
| Gram Matrix [97] | ✗ | ✗ | ✓ | 85.39 |
| TF [98] | ✗ | ✗ | ✓ | 80.69 |
| JOULE-color [100] | ✓ | ✗ | ✗ | 66.78 |
| JOULE-depth [100] | ✗ | ✓ | ✗ | 60.17 |
| JOULE-pose [100] | ✗ | ✗ | ✓ | 74.60 |
| JOULE-all [100] | ✓ | ✓ | ✓ | 78.78 |
| Huang et al. [13] | ✗ | ✗ | ✓ | 84.35 |
| Huang et al. [43] | ✗ | ✗ | ✓ | 77.57 |
| Our best (table 3) | ✗ | ✗ | ✓ | **87.3** |

Table 6. Comparison against state of the art methods using the FPHA database.

cluding differential and non-differential Laplacians which model the statistical properties of the learned graph representations, and when combined, they further enhance the performances of action recognition. Extensive experiments, conducted on standard databases (namely SBU and FPHA), show a clear gain of our design w.r.t. different handcrafted and (other) learned Laplacians, as well as the related work in skeleton-based action recognition.

## References


[1] Kumar, Sandeep, et al. "A unified framework for structured graph learning via spectral constraints." JMLR 21.22 (2020): 1-60.

[2] Khalil, Elias, et al. "Learning combinatorial optimization algorithms over graphs." In NIPS, 2017.

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

[4] B. Pasdeloup, V. Gripon, G. Mercier, D. Pastor, and M. G. Rabbat. Characterization and inference of graph diffusion processes from observations of stationary signals. IEEE Transactions on Signal and Information Processing over Networks, 2017.

[5] D. Thanou, X. Dong, D. Kressner, and P. Frossard. Learning heat diffusion graphs. IEEE Transactions on Signal and Information Processing over Networks, 3(3):484-499, 2017.

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

[7] Yaguang Li, Chuizheng Meng, Cyrus Shahabi, Yan Liu. Structure-informed Graph Auto-encoder for Relational Inference and Simulation. ICML, 2019

[8] T Kipf, E Fetaya, KC Wang, M Welling, R Zemel. Neural Relational Inference for Interacting Systems. ICML, 2018

[9] Ferran Alet, Adarsh K. Jeewajee, Maria Bauza, Alberto Rodriguez, Tomas Lozano-Perez, Leslie Pack Kaelbling. Graph Element Networks: adaptive, structured computation and memory. ICML, 2018

[10] Alet, F., Lozano-Perez, T., and Kaelbling, L. P. Modular meta-learning. In Proceedings of The 2nd Conference on Robot Learning, pp. 856-868, 2018.

[11] Luca Franceschi and Mathias Niepert and Massimiliano Pontil and Xiao He. Learning Discrete Structures for Graph Neural Networks. ICML, 2019

[12] Yu Chen, Lingfei Wu, Mohammed J. Zaki. Deep Iterative and Adaptive Learning for Graph Neural Networks. AAAI 2020 Workshop on Deep Learning on Graphs: Methodologies and Applications (AAAI DLGMA 2020)

[13] Z. Huang and L. V. Gool. A Riemannian Network for SPD Matrix Learning. In AAAI, pages 20362042, 2017.

[14] S. Yan, Y. Xiong, and D. Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. arXiv preprint arXiv:1801.07455, 2018

[15] 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.

[16] Y. Du, W. Wang, and L. Wang. Hierarchical recurrent neural network for skeleton based action recognition. In IEEE Conference on Computer Vision and Pattern Recognition, pages 11101118, 2015.

[17] J. Liu, A. Shahroudy, D. Xu, and G. Wang. Spatio-temporal LSTM with trust gates for 3D human action recognition. In European Conference on Computer Vision, pages 816833. Springer, 2016

[18] S. Song, C. Lan, J. Xing, W. Zeng, and J. Liu. An end-to-end spatio-temporal attention model for human action recognition from skeleton data. In AAAI Conference on Artificial Intelligence, volume 1, pages 42634270, 2017

[19] P. Zhang, C. Lan, J. Xing, W. Zeng, J. Xue, and N. Zheng. View adaptive recurrent neural networks for high performance human action recognition from skeleton data. arXiv preprint arXiv:1703.08274, 2017.

[20] S. Zhang, X. Liu, and J. Xiao. On geometric features for skeleton-based action recognition using multilayer lstm networks. In Winter Conference on Applications of Computer Vision, pages 148157. IEEE, 2017

[21] H. Sahbi. Kernel-based Graph Convolutional Networks. The 25th International Conference on Pattern Recognition (ICPR). IEEE, 2020.

[22] I. Lee, D. Kim, S. Kang, and S. Lee. Ensemble deep learning for skeleton-based action recognition using temporal sliding lstm networks. In IEEE International Conference on Computer Vision, pages 10121020, 2017.

[23] R. Vemulapalli, F. Arrate, and R. Chellappa. Human action recognition by representing 3D skeletons as points in a Lie group. In IEEE Conference on Computer Vision and Pattern Recognition, pages 588595, 2014

[24] T. Jiang, T. Huang and Y. Tian. Global Co-occurrence Feature Learning and Active Coordinate System Conversion for Skeleton-based Action Recognition. In Winter Conference on Applications of Computer Vision, pages 586-594. IEEE, 2020.

[25] B. Li, X. Li, Z. Zhang, and F. Wu. Spatio-temporal graph routing for skeleton-based action recognition. AAAI Conference on Artificial Intelligence, 2019

[26] Y. Wen, L. Gao, H. Fu, F. Zhang, and S. Xia. Graph CNNs with motif and variable temporal block for skeleton-based action recognition. AAAI Conference on Artificial Intelligence, 2019.

[27] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams. Convolutional networks on graphs for learning molecular fingerprints. In Conference on Neural Information Processing Sys tems, pages 22242232, 2015.

[28] M. Henaff, J. Bruna, and Y. LeCun. Deep convolutional networks on graph-structured data. arXiv preprint arXiv:1506.05163, 2015.

[29] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. IEEE Signal Processing Magazine, 30(3):8398, 2013.

[30] W. Zhu, C. Lan, J. Xing, W. Zeng, Y. Li, L. Shen, and X. Xie. Co-occurrence feature learning for skeleton based action recognition using regularized deep LSTM networks In AAAI Conference on Artificial Intelligence, volume 2, page 6, 2016.

[31] K. Yun, J. Honorio, D. Chattopadhyay, T. L. Berg, and D. Samaras. Two-person interaction detection using body pose features and multiple instance learning. In IEEE Conference on Computer Vision and Pattern Recognition Workshop, 2012.

[32] Y. Ji, G. Ye, and H. Cheng. Interactive body part contrast mining for human interaction recognition. In IEEE International Conference on Multimedia and Expo Workshop, pages 16. IEEE, 2014.

[33] W. Li, L. Wen, M. Choo Chuah, and S. Lyu. Category-blind human action recognition: A practical recognition system. In IEEE International Conference on Computer Vision, pages 44444452, 2015.

[34] L. Shi, Y. Zhang, J. Cheng, and H. Lu. Non-Local Graph Convolutional Networks for Skeleton-Based Action Recognition. CoRR, abs/1805.07694, 2018

[35] S. Yan, Y. Xiong, and D. Lin. Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. In AAAI, pages 74447452, 2018.

[36] C. Li, Z. Cui, W. Zheng, C. Xu, and J. Yang. Spatio-Temporal Graph Convolution for Skeleton Based Action Recognition. In AAAI, pages 34823489, 2018.

[37] XS. Nguyen, L. Brun, O. Lezoray and S. Bougleux. A neural network based on SPD manifold learning for skeleton-based hand gesture recognition. In IEEE International Conference on Computer Vision and Pattern Recognition, CVPR, 2019.

[38] R. Girshick. Fast R-CNN. In ICCV, pages 14401448, 2015

[39] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In CVPR, pages 770-778, June 2016.

[40] A. Mazari and H. Sahbi. Deep Temporal Pyramid Design for Action Recognition. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2019

[41] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In NIPS, pages 10971105, 2012.

[42] Z. Huang, C. Wan, T. Probst, and L. V. Gool. Deep Learning on Lie Groups for Skeleton-Based Action Recognition. In CVPR, pages 60996108, 2017.

[43] Z. Huang, J. Wu, and L. V. Gool. Building Deep Networks on Grassmann Manifolds. In AAAI, pages 32793286, 2018

[44] I. Goodfellow, Y. Bengio, and A. Courville. Deep learning. MIT press, 2016.

[45] M. Jiu and H. Sahbi. Deep kernel map networks for image annotation. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2016.

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

[47] M. Defferrard, X. Bresson, P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In NIPS, 3844-3852 (2016)

[48] W. Huang, T. Zhang, Y. Rong, J. Huang. Adaptive sampling towards fast graph representation learning. In NIPS. pp. 4558-4567 (2018)

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

[50] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, M.M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In CVPR, pp. 5115–5124 (2017)

[51] M. Gori, G. Monfardini, F. Scarselli. A new model for learning in graph domains. In IEEE IJCNN, vol. 2, pp. 729–734, 2005.

[52] A. Micheli. Neural network for graphs: A contextual constructive approach. IEEE TNN 20(3), 498-511 (2009)

[53] F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner, G. Monfardini. The graph neural network model. IEEE TNN 20(1), 61–80, 2008.

[54] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, P.S. Yu. A comprehensive survey on graph neural networks. arXiv:1901.00596 (2019).

[55] W. Hamilton, Z. Ying, J. Leskovec. Inductive representation learning on large graphs. In NIPS. pp. 1024–1034 (2017)

[56] H. Sahbi. Learning Connectivity with Graph Convolutional Networks. The 25th International Conference on Pattern Recognition (ICPR). IEEE, 2020.

[57] R. Levie, F. Monti, X. Bresson, M.M. Bronstein. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. IEEE Transactions on Signal Processing 67(1), 97–109 (2018)

[58] J. Chen, T. Ma, C. Xiao. Fastgcn: fast learning with graph convolutional networks via importance sampling. arXiv preprint arXiv:1801.10247 (2018)

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

[60] W. Huang, T. Zhang, Y. Rong, J. Huang. Adaptive sampling towards fast graph representation learning. In NIPS. pp. 4558-4567 (2018)

[61] A. Mazari and H. Sahbi. MLGCN: Multi-Laplacian Graph Convolutional Networks for Human Action Recognition. BMVC. 2019.

[62] D. Slepian. Some comments on Fourier analysis, uncertainty and modeling. In Society for Industrial and Applied Mathematics (SIAM review), 1983

[63] F. Chung. Spectral graph theory. American Mathematical Soc.. 1997.

[64] J. Atwood, D. Towsley. Diffusion-convolutional neural networks. In NIPS, pp. 1993–2001 (2016)

[65] H. Gao, Z. Wang, S. Ji. Large-scale learnable graph convolutional networks. In the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 1416–1424. ACM (2018)

[66] M. Niepert, M. Ahmed, K. Kutzkov. Learning convolutional neural networks for graphs. In ICML, pp. 2014-2023 (2016)

[67] O. Quentin and H. Sahbi. Learning attribute representations for remote sensing ship category classification. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 10.6 (2017): 2830-2840.

[68] A. Dutta and H. Sahbi. High order stochastic graphlet embedding for graph-based pattern recognition. arXiv preprint arXiv:1702.00156 (2017).

[69] H. Sahbi. Imageclef annotation with explicit context-aware kernel maps. International Journal of Multimedia Information Retrieval 4.2 (2015): 113-128.

[70] H. Sahbi. CNRS-TELECOM ParisTech at ImageCLEF 2013 Scalable Concept Image Annotation Task: Winning Annotations with Context Dependent SVMs. CLEF (Working Notes). 2013.

[71] S. Thiemert, H. Sahbi and M. Steinebach. Using entropy for image and video authentication watermarks. Security, Steganography, and Watermarking of Multimedia Contents VIII. Vol. 6072. International Society for Optics and Photonics, 2006.

[72] J. Zhang, X. Shi, J. Xie, H. Ma, I. King, D.Y. Yeung. Gaan: Gated attention networks for learning on large and spatiotemporal graphs. arXiv:1803.07294 (2018)

[73] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst, Learning Laplacian matrix in smooth graph signal representations, IEEE Trans. Signal Processing, vol. 64, no. 23, pp. 6160-6173, 2016.

[74] S.P. Chepuri, S. Liu, G. Leus, and A.O. Hero, Learning sparse graphs under smoothness prior, in Proc. of ICASSP, 2017, pp. 6508-6512.

[75] H.E. Egilmez, E. Pavez, and A. Ortega, Graph learning from data under structural and laplacian constraints, arXiv preprint arXiv:1611.05181,2016.

[76] V. Kalofolias, How to learn a graph from smooth signals, in Proc. of the conf. on Artificial Intelligence and Statistics, 2016, pp. 920-929.

[77] H. Sahbi, P. Etyngier, J-Y. Audibert and Keriven, R. Manifold learning using robust graph laplacian for interactive image search. In IEEE Conference on Computer Vision and Pattern Recognition (pp. 1-8), 2008.

[78] M. Jiu and H. Sahbi. Semi supervised deep kernel design for image annotation. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2015.

[79] M. Belkin and P. Niyogi. Lapl eigenmaps for dimensionality reduction and data representation. Neural computation 15.6 (2003): 1373-1396.

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

[81] S.I. Daitch, J.A. Kelner, and D.A. Spielman, Fitting a graph to vector data, in Proc. of ICML, 2009, pp. 201-208.

[82] S. Sardellitti, S. Barbarossa, and P. Di Lorenzo, Graph topology inference based on transform learning, in Proc. of the Global Conf. on Signal and Information Processing, 2016, pp. 356-360.

[83] B. Le Bars, P. Humbert, L. Oudre and A. s Kalogeratos. Learning Laplacian Matrix from Bandlimited Graph Signals. In ICASSP, 2019

[84] S. Sardellitti, S. Barbarossa, and P. Di Lorenzo. Graph topology inference based on sparsifying transform learning. IEEE TSP, 67(7), 2019.

[85] P. Vo and H. Sahbi. Transductive kernel map learning and its application to image annotation. BMVC. 2012.

[86] W. Li, Y. Lu, K. Zheng, H. Liao, C. Lin, J. Luo & Miao, S. Structured Landmark Detection via Topology-Adapting Deep Graph Learning. arXiv preprint arXiv:2004.08190. 2020

[87] 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.

[88] K. Yun, J. Honorio, D. Chattopadhyay, T.L. Berg, and D. Samaras. Two-person interaction detection using body pose features and multiple instance learning. In IEEE Conference on Computer Vision and Pattern Recognition Workshop, 2012.

[89] M. Meshry, M.E. Hussein, and M. Torki. "Linear-time online action detection from 3d skeletal data using bags of gesturelets." 2016 IEEE Winter Conference on Applications of Computer Vision (WACV). IEEE, 2016.

[90] M. Maghoumi, JJ. LaViola Jr. DeepGRU: Deep Gesture Recognition Utility. In arXiv preprint arXiv:1810.12514, 2018

[91] C. Feichtenhofer, A. P., and A. Zisserman. Convolutional Two-Stream Network Fusion for Video Action Recognition. CVPR, pages 1933-1941, 2016.

[92] E. Ohn-Barand and M.M. Trivedi. Hand Gesture Recognitionin Real Time for Automotive Interfaces: A Multimodal Vision- Based Approach and Evaluations. IEEE Transactions on Intelligent Transportation Systems, 15(6):2368-2377, 2014.

[93] L. Wang and H. Sahbi. Nonlinear cross-view sample enrichment for action recognition. European Conference on Computer Vision. Springer, 2014.

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

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

[96] M. Zanfir, M. Leordeanu, and C. Sminchisescu. The Mov- ing Pose: An Efficient 3D Kinematics Descriptor for Low- Latency Action Recognition and Detection. In ICCV, pages 2752-2759, 2013.

[97] X. Zhang, Y. Wang, M. Gou, M. Sznaier, and O. Camps. Efficient Temporal Sequence Comparison and Classification Using Gram Matrix Embeddings on a Riemannian Manifold. In CVPR, pages 4498-4507, 2016.

[98] G. Garcia-Hernando and T.-K. Kim. Transition Forests: Learning Discriminative Temporal Transitions for Action Recognition. In CVPR, pages 407-415, 2017.

[99] L. Wang and H. Sahbi. Bags-of-daglets for action recognition. IEEE International Conference on Image Processing (ICIP), 2014.

[100] J. Hu, W. Zheng, J. Lai, and J. Zhang. Jointly Learning Het- erogeneous Features for RGB-D Activity Recognition. In CVPR, pages 5344-5352, 2015.

[101] D.P. Kingma, and J. Ba. ”Adam: A method for stochastic optimization.” arXiv preprint arXiv:1412.6980 (2014).

[102] R. Ando and T. Zhang. Learning on graph with Laplacian regularization. Advances in neural information processing systems, 19, 25-32 (2006).

[103] A. Kacem, M. Daoudi, B. Ben Amor, S. Berretti, J-Carlos. Alvarez-Paiva. A Novel Geometric Framework on Gram Matrix Trajectories for Human Behavior Understanding. IEEE Transactions on Pattern Analysis and Machine Intelligence, 28 September 2018

[104] J. Liu, G. Wang, L. Duan, K. Abdiyeva, and A. C. Kot. Skeleton-based human action recognition with global context-aware attention lstm networks. IEEE Transactions on Image Processing, 27(4):15861599, April 2018