HAL open science

MLGCN: Multi-Laplacian Graph Convolutional Networks for Human Action Recognition

Ahmed Mazari, Hichem Sahbi

To cite this version:

Ahmed Mazari, Hichem Sahbi. MLGCN: Multi-Laplacian Graph Convolutional Networks for Human Action Recognition. The British Machine Vision Conference (BMVC), Sep 2019, Cardiff, United Kingdom. hal-03089634

HAL Id: hal-03089634

https://hal.science/hal-03089634v1

Submitted on 28 Dec 2020

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.

MLGCN: Multi-Laplacian Graph Convolutional Networks for Human Action Recognition

Ahmed Mazari
ahmed.mazari@lip6.fr
Sorbonne University, UPMC, CNRS, LIP6, F-75005 Paris, France

Hichem Sahbi
hichem.sahbi@lip6.fr
Sorbonne University, UPMC, CNRS, LIP6, F-75005 Paris, France

Abstract

Convolutional neural networks are nowadays witnessing a major success in different pattern recognition problems. These learning models were basically designed to handle vectorial data such as images but their extension to non-vectorial and semi-structured data (namely graphs with variable sizes, topology, etc.) remains a major challenge, though a few interesting solutions are currently emerging.
In this paper, we introduce MLGCN; a novel spectral Multi-Laplacian Graph Convolutional Network. The main contribution of this method resides in a new design principle that learns graph-laplacians as convex combinations of other elementary laplacians – each one dedicated to a particular topology of the input graphs. We also introduce a novel pooling operator, on graphs, that proceeds in two steps: context-dependent node expansion is achieved, followed by a global average pooling; the strength of this two-step process resides in its ability to preserve the discrimination power of nodes while achieving permutation invariance. Experiments conducted on SBU and UCF-101 datasets, show the validity of our method for the challenging task of action recognition.

1 Introduction

Video action recognition is a major task in computer vision which consists in classifying sequences of frames into categories (or classes) of actions. This task is known to be challenging due to the intrinsic properties (appearance and motion) of moving objects and also their extrinsic acquisition conditions (occlusions, background clutter, camera motion, illumination, length/resolution, etc.). Most of the existing action recognition methods are based on machine learning [19, 20, 21, 22, 23, 54, 57, 80]; their general recipe consists in extracting (handcrafted or learned) features prior to classifying them using inference techniques such as kernel methods and deep networks [3, 5, 9, 16, 17, 18, 24, 25, 26, 31, 59, 84].

Among the machine learning techniques – for action recognition – those based on deep networks are particularly performant; successful methods include two-stream 2D convolutional neural networks (CNNs) [17], two-stream 3D CNNs and simple 3D CNNs [18]. However, and beside being data-hungry, these models rely on a strong assumption that videos are described as vectorial data; in other words, these methods assume that videos come only in

© 2019. The copyright of this document resides with its authors.
It may be distributed unchanged freely in print or electronic forms.

the form of regular (2D or 3D) grids. This assumption may not hold in practice: on the one hand, one may consider moving objects as constellations of interacting body parts (such as 2D/3D skeletons or joints in human actions) and this requires processing only these joints without taking into account *holistically* cluttered background or other parts in the scenes. On the other hand, moving objects may be occluded with spurious details which are not necessarily related to the moving object parts. Hence, for these particular settings, graph convolutional networks (GCNs) [62] are rather more appropriate where nodes, in these models, capture object parts and links their spatio-temporal interactions.

Early GCNs are targeted to graphs with known/fixed topology[1] (fixed number of nodes/edges, constant degree, etc.) [1, 4]; in existing solutions pixels are considered as nodes and edges connect neighboring pixels. Despite their relative success for some pattern classification tasks including optical character recognition (on widely used benchmarks such as MNIST), these methods do not straightforwardly extend to general graphs with arbitrary topological characteristics (variable number of nodes/edges, heterogeneous degrees, etc.) and this limits their applicability to other challenging tasks such as action recognition. Recent attempts, to extend these methods to action recognition [27, 58, 60], include [27] which models connectivity of moving joints in videos using graphs where nodes correspond to joints (described by spatial coordinates and their likelihoods) and edges characterize their spatio-temporal interactions. One of the drawbacks of these extensions resides in the limited representational power of joints and also the difficulty in achieving permutation invariance; in other words, parsing and describing joints while being invariant to arbitrary reordering of objects especially for highly complex scenes with multiple interacting objects/persons. From the machine learning point of view, GCN operates either directly in the spatial domain [8, 10, 11, 28, 32, 33, 34, 35, 37, 38, 56, 70, 74, 77] or require a preliminary step of spectral decomposition of graphs using Fourier basis [12, 13, 75, 76] prior to achieve convolution [1, 2, 4, 6, 7, 36, 63, 64, 65]. While graph convolution in the spectral domain is well defined, its success heavily relies on the choice of the laplacian operators [78] that capture the topology of the manifolds enclosing data. These laplacians, in turn, depend on many hyper-parameters which are difficult to set using tedious cross-validation especially when training GCNs on large-scale datasets.

In this paper, we address the aforementioned issues (mainly laplacian design in GCNs and permutation invariance) for the particular task of action recognition. Our solution achieves convolution in the spectral domain using a new design principle that considers a convex combination of several laplacian operators; each laplacian is dedicated to a particular (possible) topology of our graphs. We also introduce a novel context-dependent pooling operator that proceeds in two steps: node features are first expanded with their contexts and then globally averaged; the strength of this two-step pooling process resides in its ability to preserve/enhance the discrimination power of node representations while achieving permutation invariance. The validity of these contributions is corroborated through extensive experiments, in action recognition, using the challenging SBU-skeleton and UCF-101 datasets.

2 Graph Construction

In this section, we briefly describe the video processing used to build our input graphs. This step consists in extracting and grouping joints (a.k.a keypoints) into trajectories prior to

[1] as 2D regular grids (see also [14, 56]).

modeling their spatio-temporal interactions with graphs.

Given a raw video, skeletons are obtained by detecting human joints in successive frames using the state of the art human pose extractor [15][2]; as these keypoints are labeled (see Fig. 1), their trajectories are extracted by simply tracking keypoints with the same labels. Considering a finite collection of trajectories, we build an adjacency graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where each node $v \in \mathcal{V}$ corresponds to a labeled trajectory and an edge $(v, v') \in \mathcal{E}$ exists between two nodes iff the underlying trajectories are spatially neighbors. Each trajectory (i.e., node in $\mathcal{G}$) is described by aggregating motion and appearance streams as shown subsequently.

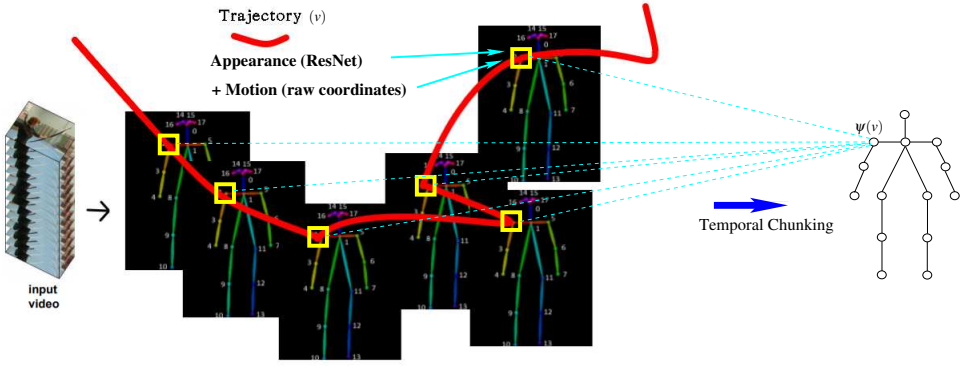

Figure 1: This figure shows the whole keypoint extraction, tracking and description process on motion and appearance streams (see also the detailed protocol in the supplementary material [37]).

Motion stream. Considering a video as a sequence of skeletons, we process the underlying trajectories using *temporal chunking*: first we split the total duration of a video into C equally-sized temporal chunks ($C = 4$ in practice), then we assign the keypoint coordinates of a given trajectory v to the C chunks (depending on their time stamps) prior to concatenate the averages of these chunks and this produces the description of v denoted as $\psi(v)$. Hence, trajectories, with similar keypoint coordinates but arranged differently in time, will be considered as very different. Note that beside being compact and discriminant (as shown later in table 3(c)), 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 descriptions that preserve the temporal structure of trajectories while being *frame-rate* and *duration* agnostic.

Appearance stream. Similarly to motion, we also describe each trajectory using appearance features. First, we apply ResNet [73] frame-wise[3] in order to collect convolutional features associated to different keypoints (see again Fig. 1), then we aggregate those convolutional features through trajectories using temporal chunking as described above for motion stream.

[2]This processing is only reserved to raw video datasets (including UCF [39]) while for other databases, such as SBU [33], skeletons are already available.

[3]We consider a local neighborhood around each keypoint in order to extract these convolutional features.

3 Multi-Laplacian Convolutional Networks

Given a collection of videos, we describe each one using a graph $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$ as shown in section 2. For each node $v \in \mathcal{V}_i$, we extract two feature vectors, denoted $\psi_m(v)$, $\psi_a(v)$, respectively corresponding to motion and appearance streams of v. We also define a similarity between nodes in $\mathcal{V}_i$ as $k_m(v,v') = \exp(-\|\psi_m(v) - \psi_m(v')\|_2^2/\sigma_m)$, here σ_m is the scale of the gaussian similarity and $\|.\|_2$ is the ℓ_2 norm. Similarly, we define $k_a(v,v')$ using appearance features. In the remainder of this paper, unless explicitly mentioned, we denote a given graph $\mathcal{G}_i$ simply as $\mathcal{G}$. We also denote motion and appearance features $\psi_m(v)$, $\psi_a(v)$ as $\psi(v)$, scales σ_m, σ_a as σ, and similarities $k_m(v,v')$, $k_a(v,v')$ as $k(v,v')$.

The goal is to design a GCN that returns the representation and the classification of a given graph. This includes *a novel design of laplacian convolution and pooling* on graphs as shown subsequently.

3.1 Spectral graph convolution at a glance

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 normalized, unormalized or random walk laplacians respectively defined as $\mathbf{L} = \mathbf{I}_n - \mathbf{D}^{-1/2}\,\mathbf{A}\,\mathbf{D}^{-1/2}$, $\mathbf{L} = \mathbf{D} - \mathbf{A}$ and $\mathbf{L} = \mathbf{D}^{-1}\mathbf{A}$ where $\mathbf{I}_n$ is an $n \times n$ identity matrix, $\mathbf{A}$ is the affinity matrix built as $[\mathbf{A}]_{vv'} = \mathbb{1}_{\{(v,v')\in\mathcal{E}\}}$ or by using the gaussian similarity $k(.,.)$ as $[\mathbf{A}]_{vv'} = \mathbb{1}_{\{(v,v')\in\mathcal{E}\}}.k(\psi(v), \psi(v'))$ and $\mathbf{D}$ a diagonal degree matrix with each diagonal entry $[\mathbf{D}]_{vv} = \sum_{v'}[\mathbf{A}]_{vv'}$. Considering the eigen-decomposition of $\mathbf{L}$ as $\mathbf{U}\Lambda\mathbf{U}'$ with $\mathbf{U}$, Λ being respectively the matrix of its eigenvectors (graph Fourier modes) and the diagonal matrix of its non-negative eigenvalues, spectral graph convolution is a well defined operator (see for instance [1]) 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 the graph signal $\psi(\mathcal{V}) \in \mathbb{R}^{n\times p}$ is $(\psi \star g_\theta)(\mathcal{V}) = \mathbf{U}\; g_\theta(\Lambda)\; \mathbf{U}'\; \psi(\mathcal{V})$; here g_θ 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 [1]

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

with K fixed and $\theta = (\theta_1 \dots \theta_K)' \in \mathbb{R}^K$ being its learned convolutional filter parameters; in practice, we consider a rescaled version of the laplacian (i.e., $2\mathbf{L}/\lambda_{max} - \mathbf{I}_n$ instead of $\mathbf{L}$ with λ_{max} being its largest eigenvalue). In the above equation, T_k is the k-th order Chebyshev polynomial recursively defined as $T_k(\mathbf{L}) = 2\mathbf{L}\; T_{k-1}(\mathbf{L}) - T_{k-2}(\mathbf{L})$, with $T_k(\mathbf{L}) \in \mathbb{R}^{n\times n}$ and $T_0 = \mathbf{I}$, $T_1 = \mathbf{L}$ (for more details see again [1]).

3.2 Multi-Laplacian design

The success of the aforementioned convolutional process is highly dependent on the *relevance* of the used laplacian, which in turn depends on the appropriate choice of the affinity matrix of the graph and its hyper-parameters. Hence, knowing a priori which parameter to choose could be challenging and usually relies on the tedious cross-validation.

Our alternative contribution in this paper aims at designing convolutional laplacian operators while learning the topological structure of the input graphs (characterized by their laplacians). Starting from different *elementary* laplacians[4] associated to multiple settings (for instance, by varying the scale σ of the gaussian similarity $k(.,.)$ and the laplacians), we train a *multiple laplacian* as a deep nonlinear combination of multiple elementary laplacians. Fig. 2 shows our learning framework with d-layers in the multi-laplacian; for each layer $\ell+1$ ($\ell \in \{0,\ldots,d-1\}$) and its associated unit $p \in \{1,\ldots,n_{\ell+1}\}$, a laplacian (denoted $\mathbf{L}_p^{\ell+1}$) is recursively defined as

$$\mathbf{L}_p^{\ell+1} = g\left(\sum_{q=1}^{n_\ell} \mathbf{w}_{q,p}^{\ell}\, \mathbf{L}_q^{\ell}\right), \tag{2}$$

where g is a nonlinear activation function (see details in section 3.3), n_ℓ is the number of units in layer ℓ and $\{\mathbf{w}_{q,p}^{\ell}\}_q$ are the (learned) weights associated to $\mathbf{L}_p^{\ell+1}$. For any given graph $\mathcal{G}$, a tensor of multiple elementary laplacians $\{\mathbf{L}_q^1\}_q$ (associated to different combinations of $\{\sigma\}$ and standard laplacians namely unormalized, normalized, random walk, etc.) on $\mathcal{G}$ is considered as an input to our deep network. These elementary laplacians are then forwarded to the subsequent intermediate layer resulting into n_2 multiple laplacians through the nonlinear combination of the previous layer, etc. The final laplacian $\mathbf{L}_1^d$ is a highly nonlinear combination of elementary laplacians. We notice that the deep laplacian network in essence is a multi-layer perceptron (MLP), with nonlinear activation functions which is fed (together with the graph signal $\psi(\mathcal{V})$) as input in order to achieve convolution (see Fig. 2). Hence, we can use standard backpropagation in order to optimize the parameters of both the MLP and the GCN networks. Let J denote the loss function associated to our classification problem (namely cross-entropy); starting from the gradients of this loss J w.r.t the final softmax output, we use the chain rule in order to backpropagate the gradients w.r.t different layers and parameters (fully connected and convolutional layers as well as the MLP of the multi-laplacians), and to update these parameters accordingly using gradient descent.

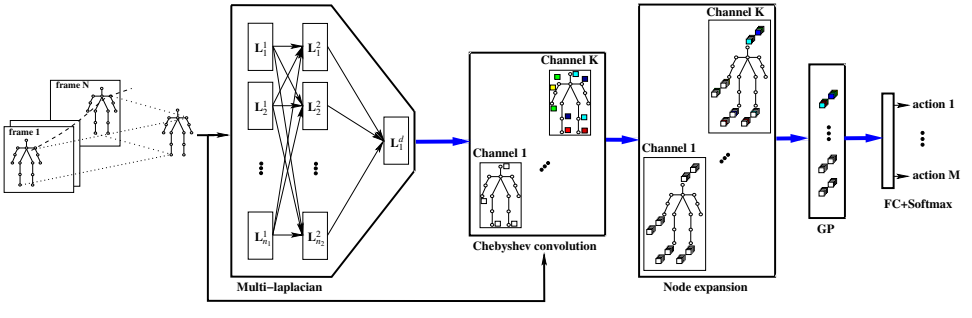

Figure 2: This figure shows the architecture of our multi-laplacian graph convolutional network (ML-GCN). First, multiple elementary laplacians (associated to $\mathcal{G} = (\mathcal{V},\mathcal{E})$) and graph signal $\psi(\mathcal{V})$ are fed as input to an MLP in order to learn the best combination of laplacians. Then, Chebyshev decomposition is achieved using the learned multi-laplacian in order to perform graph convolution, followed by node expansion and global average pooling prior to softmax classification **(better to zoom the pdf)**.

[4] also referred to as single or individual laplacians.

3.3 Activation functions and optimization

We consider two activation functions g in Eq. (2): ReLU and leaky ReLU [81, 82, 83]. Note that only leaky ReLU provides negative entries in the learned laplacians and both of these activations allow learning *conditionally* positive definite (c.p.d) laplacian matrices. In what follows, we discuss the sufficient conditions about the choices of the elementary input laplacians, the parameters $\{\mathbf{w}_{q,p}^{\ell}\}$ and the activation functions that guarantee this c.p.d property.

Definition 1 (conditionally positive definite laplacians) *A laplacian matrix* $\mathbf{L}$ *is conditionally positive definite, iff* $\forall c_1,\ldots,c_n \in \mathbb{R}$ *(***with** $\sum_{i=1}^{n} c_i = 0$*),* $\sum_{i,j} c_i c_j \mathbf{L}_{ij} \geq 0$.

From the above definition, it is clear that any positive definite laplacian is also c.p.d. The converse is not true, however c.p.d is a weaker (but sufficient) condition in order to derive positive definite laplacians (see following propositions).

Proposition 1 (Berg et al.[85]) *Consider* $\mathbf{L}_{i,j}$ *as an entry of a matrix* $\mathbf{L}$ *and define* $\hat{\mathbf{L}}$ *with*

$$\hat{\mathbf{L}}_{i,j} \quad = \quad \mathbf{L}_{i,j} - \mathbf{L}_{i,n+1} - \mathbf{L}_{n+1,j} + \mathbf{L}_{n+1,n+1} \tag{3}$$

Then, $\hat{\mathbf{L}}$ *is positive definite if and only if* $\mathbf{L}$ *is c.p.d.*

Proof 1 *See the supplementary material [82]. Now we derive our main result:*

Proposition 2 *Provided that the input elementary laplacians* $\{\mathbf{L}_q^1\}_q$ *are c.p.d, and* $\{\mathbf{w}_{q,p}^{\ell}\}_{p,q,\ell}$ *belong to the positive orthant of the parameter space, any combination* $g(\sum_q \mathbf{w}_{q,p}^{\ell}\, \mathbf{L}_q^{\ell})$*, with* g *equal to ReLU or leaky ReLU, is also c.p.d.*

Proof 2 *See appendix.*

From proposition (2), provided that i) the elementary laplacians are c.p.d, ii) the activation function g preserves the c.p.d (as ReLU and leaky-ReLU) and iii) weights $\{\mathbf{w}_{q,p}^{\ell}\}$ are positive, all the resulting multiple laplacians in Eq. 2 will also be c.p.d and admit equivalent positive definite laplacians (following proposition 1), and thereby spectral graph convolution can be achieved. Note that conditions (i) and (ii) are satisfied by construction while condition (iii) requires adding equality and inequality constraints to Eq. 2, i.e., $\mathbf{w}_{q,p}^{\ell} \in [0,1]$ and $\sum_q \mathbf{w}_{q,p}^{\ell} = 1$. In order to implement these constraints, we consider a reparametrization in Eq. 2 as $\mathbf{w}_{q,p}^{\ell} = f(\hat{\mathbf{w}}_{q,p}^{\ell})/\sum_q f(\hat{\mathbf{w}}_{q,p}^{\ell})$ for some $\{\hat{\mathbf{w}}_{q,p}^{\ell}\}$ with f being strictly monotonic real-valued (positive) function and this allows free settings of the parameters $\{\hat{\mathbf{w}}_{q,p}^{\ell}\}$ during optimization while guaranteeing $\mathbf{w}_{q,p}^{\ell} \in [0,1]$ and $\sum_q \mathbf{w}_{q,p}^{\ell} = 1$. During backpropagation, the gradient of the loss J (now w.r.t $\hat{\mathbf{w}}$'s) is updated using the chain rule as

$$\frac{\partial J}{\partial \hat{\mathbf{w}}_{q,p}^{\ell}} \quad = \quad \frac{\partial J}{\partial \mathbf{w}_{q,p}^{\ell}} \cdot \frac{\partial \mathbf{w}_{q,p}^{\ell}}{\partial \hat{\mathbf{w}}_{q,p}^{\ell}} \qquad \text{with} \quad \frac{\partial \mathbf{w}_{q,p}^{\ell}}{\partial \hat{\mathbf{w}}_{q,p}^{\ell}} = \frac{f'(\hat{\mathbf{w}}_{q,p}^{\ell}) f(\sum_{r\neq q} \hat{\mathbf{w}}_{r,p}^{\ell})}{(f(\hat{\mathbf{w}}_{q,p}^{\ell}) + f(\sum_{r\neq q} \hat{\mathbf{w}}_{r,p}^{\ell}))^2}, \tag{4}$$

in practice $f(.) = \exp(.)$ and $\frac{\partial J}{\partial \mathbf{w}_{q,p}^{\ell}}$ is obtained from layerwise gradient backpropagation (as already integrated in standard deep learning tools including PyTorch and TensorFlow). Hence, $\frac{\partial J}{\partial \hat{\mathbf{w}}_{q,p}^{\ell}}$ is obtained by multiplying the original gradient $\frac{\partial J}{\partial \mathbf{w}_{q,p}^{\ell}}$ by $\frac{\exp(\sum_r \hat{\mathbf{w}}_{r,p}^{\ell})}{(\exp(\hat{\mathbf{w}}_{q,p}^{\ell}) + \exp(\sum_{r\neq q} \hat{\mathbf{w}}_{r,p}^{\ell}))^2}$.

3.4 Pooling

If pooling on regular grids (or vectorial data in general) is well defined, it is not the case for graphs [79]. As a consequence, most of GCN architectures do not include pooling layers in their architectures [9, 29] excepting a few attempts which try to incorporate pooling in a non explicit way using multi-level graph coarsening (i.e., by reducing graphs by a factor of two at each level and describing each node by the average or the max of its descendants [1, 30] or by using clustering [69, 72] and reordering [67, 68, 70, 71]). For highly irregular graphs (e.g., with heterogeneous degrees), this graph coarsening process usually results into imbalanced hierarchical representations and this substantially affects the accuracy of the learned graph representations. In practice, existing methods (for instance [1]) add fake nodes in the input graphs in order to rebalance the coarsening process. However, fake nodes are spurious and this may lead to contaminated graph representations after coarsening. Besides, this pooling process is not invariant to node permutations and node reordering (based on automorphisms) cannot guarantee permutation invariance for general and irregular graphs.

In this work, we consider an alternative solution in order to achieve pooling. Our method relies on two steps: an expansion-step is first achieved at the node level followed by a global average pooling in order to achieve permutation invariance. Note that the first step (expansion) is necessary in order to generate high dimensional (and sparse) node representations and hence preserve the discrimination power of nodes before applying the second step of global average pooling. Put differently, without expansion, average pooling achieves permutation invariance but dilutes node information and this results into less discriminant graph representations as shown in experiments (see also [90]).

Considering $\mathcal{N}_r(v)$ as the set of r-hop neighbors of a given node $v \in \mathcal{V}$ and $\mathcal{N}_r(v) = \cup_{l=1}^{L} \mathcal{N}_r^l(v)$ as the union of L subsets[5], the expansion of v is defined as

$$\phi(v) \leftarrow \left((\psi \star g_\theta)(v), \quad \frac{1}{|\mathcal{N}_r^1(v)|} \sum_{v' \in \mathcal{N}_r^1(v)} (\psi \star g_\theta)(v'), \ldots, \frac{1}{|\mathcal{N}_r^L(v)|} \sum_{v' \in \mathcal{N}_r^L(v)} (\psi \star g_\theta)(v') \right). \tag{5}$$

For a large and fine-grained neighborhood system $\mathcal{N}_r(v) = \cup_{l=1}^{L} \mathcal{N}_r^l(v)$ (i.e., $r \geq 1$ and $L \gg 1$), the expansion $\phi(v)$ takes into account not only the immediate neighbors of v but also a large extent and this results into high dimensional, sparse and discriminating representations. Finally, a global average pooling is performed (as $\sum_{v \in \mathcal{V}} \phi(v)$) to achieve permutation invariance prior to the softmax fully connected classification layer (see again Fig. 2).

4 Experiments

We evaluate the performance of our multi-laplacian graph convolutional networks (MLGCN) on the challenging task of action recognition, using two standard datasets: SBU kinect [88] and UCF-101 [89]. SBU is an interaction dataset acquired (under relatively well controlled conditions) using the Microsoft Kinect sensor; it includes in total 282 video sequences belonging to 8 categories: "approaching", "departing", "pushing", "kicking", "punching", "exchanging objects", "hugging", and "hand shaking". In contrast, UCF-101 is larger and more challenging; it includes 13,320 video shots belonging to 101 categories with variable duration, poor frame resolution, viewpoint and illumination changes, occlusion, cluttered

[5] In practice, each subset $\mathcal{N}_r^l(v)$ includes only nodes with labels equal to l (see again node labels in Fig. 1).

background and eclectic content ranging from multiple and highly interacting individuals to single and completely passive ones. In all these experiments, we use the same evaluation protocols as the ones suggested in [88, 89] (i.e., split2 for UCF-101 and train-test split for SBU) and we report the average accuracy over all the classes of actions.

		Binary	Binary × Gaussian													Multi-lap
			$10^{-6}\sigma$	$10^{-5}\sigma$	$10^{-4}\sigma$	$10^{-3}\sigma$	$10^{-2}\sigma$	$10^{-1}\sigma$	σ	10σ	$10^{2}\sigma$	$10^{3}\sigma$	$10^{4}\sigma$	$10^{5}\sigma$	$10^{6}\sigma$	
Unormalized	$k=1$	93.00	92.32	92.32	92.32	92.32	92.32	92.32	92.32	92.32	92.30	92.30	92.30	92.30	92.30	93.41
	$k=4$	89.25	88.87	88.87	88.87	88.87	88.87	88.87	88.87	88.87	88.87	88.86	88.86	88.86	88.86	90.07
	$k=32$	86.00	86.31	86.31	86.31	86.31	84.31	86.31	86.31	86.32	86.32	86.32	86.32	86.32	86.32	86.91
Normalized	$k=1$	93.00	92.28	92.28	92.28	92.28	92.28	92.28	92.26	92.26	92.26	92.26	92.26	92.28	92.28	93.49
	$k=4$	90.00	89.36	89.36	89.36	89.36	89.36	89.36	89.36	89.38	89.38	89.39	89.37	89.37	89.37	91.49
	$k=32$	88.00	88.31	88.31	88.31	88.31	88.31	88.31	88.31	88.32	88.32	88.32	88.32	88.32	88.32	89.21
Random w	$k=1$	93.00	92.05	92.05	92.06	92.05	92.05	92.05	92.05	92.09	92.09	92.09	92.06	92.06	92.06	93.46
	$k=4$	96.00	94.06	94.06	94.06	94.00	94.00	94.00	94.01	94.00	94.01	94.00	94.00	94.00	94.00	96.31
	$k=32$	96.00	94.03	94.03	94.03	94.03	94.03	94.03	94.03	94.03	94.02	94.02	94.02	94.02	94.02	96.29
Multi-lap		97.15	94.61	94.58	94.61	94.63	94.63	94.63	94.62	94.63	94.63	94.63	94.63	94.63	94.63	**98.6**

Table 1: Performances on SBU for different elementary laplacians (normalized, unormalized and random walk) and their marginal and total combinations using MLGCN (note that our expansion+GP is used for pooling). In this table, "binary" means that $\mathbf{A}^k$ is used to build the elementary laplacian while "binary × gaussian" means that "$\mathbf{A}^k\times$ gaussian similarity" is used instead; for each graph $\mathcal{G}$, the scale σ of the gaussian similarity is taken as the average distance between node features in $\mathcal{G}$. See also table 5 in supplementary material including results *without expansion*.

		Binary	Binary × Gaussian													Multi-lap
			$10^{-6}\sigma$	$10^{-5}\sigma$	$10^{-4}\sigma$	$10^{-3}\sigma$	$10^{-2}\sigma$	$10^{-1}\sigma$	σ	10σ	$10^{2}\sigma$	$10^{3}\sigma$	$10^{4}\sigma$	$10^{5}\sigma$	$10^{6}\sigma$	
Unormalized	$k=1$	55.32	50.67	50.67	50.67	50.68	50.70	50.70	50.70	50.71	50.72	50.72	50.72	50.70	50.70	56.55
	$k=4$	59.23	55.22	55.22	55.22	55.22	55.20	55.20	55.20	54.95	54.96	54.95	54.98	55.00	54.98	60.05
	$k=32$	55.10	52.05	52.05	52.05	52.05	52.11	52.11	52.11	52.11	52.11	52.06	52.06	52.06	52.08	56.48
Normalized	$k=1$	55.6	50.78	50.77	50.27	50.42	50.40	50.42	50.42	50.42	50.42	50.42	50.42	50.42	50.42	56.80
	$k=4$	59.45	55.32	55.35	55.35	55.00	55.00	54.60	54.60	54.60	54.60	54.60	54.60	54.60	54.60	60.35
	$k=32$	55.25	51.19	51.19	51.19	49.78	49.79	49.79	49.79	49.79	49.78	49.78	49.77	49.77	49.77	56.52
Random w	$k=1$	60.09	58.00	58.00	57.98	58.00	58.00	58.00	58.00	58.01	57.95	57.95	57.95	57.92	57.94	60.85
	$k=4$	61.63	58.05	58.05	58.05	58.05	58.05	58.02	58.02	58.02	57.98	57.98	57.98	57.98	58.02	61.90
	$k=32$	60.23	58.02	58.02	58.02	58.02	58.01	58.02	58.02	58.01	57.95	57.95	57.95	57.92	57.92	60.9
Multi-lap		62.00	58.24	58.16	58.14	58.14	58.14	58.15	58.15	58.13	58.14	58.16	58.18	58.15	58.17	**63.27**

Table 2: Performance on UCF; see caption of table 1 for the settings. See also table 6 in supplementary material including results *without expansion*.

We trained our MLGCN for 150 epochs on UCF-101 (and 40 on SBU) using the PyTorch SGD optimizer and we set the learning rate to 0.0006 (decayed by a factor 0.1 after 100 epochs) for UCF-101 and 0.7 for SBU. We set the batch size to 30 and the Chebyshev order K to 4 using grid search and cross validation. All these experiments are run on GPUs; Tesla P100 (with 16 Go) for UCF-101 and Titan X Pascal (with 12 Go) for SBU. No data augmentation is achieved. Tables 1 and 2 show a comparison of action recognition performances, using MLGCN against different baselines involving individual laplacians (normalized, unormalized, random walk built on top of different affinity matrices and scale parameters). In these tables, we show the results using expansion and global average pooling (GP). We also show in table 3(a-c) the results for (i) different pooling strategies (no-pooling, only GP, feature propagation [87] and feature propagation+GP), (ii) various multi-laplacian depths and activation functions[6] and (iii) different input graph descriptions (for SBU). From all these results, we observe a clear and a consistent gain of MLGCN w.r.t all the individual laplacian settings; this gain is further amplified when using "expansion+GP" with a large spatial extent and a fine-grained neighborhood system $\mathcal{N}_r(v) = \cup_{l=1}^{L}\mathcal{N}_r^l(v)$ (i.e., $r \geq 1$ and

[6] As shown in table 3(b), performances improve/stabilize very quickly, as the depth increases, since the size of the training set is limited compared to the large number of training parameters in the MLP of the multi-laplacian. These performances are consistently better when using leaky ReLU (compared to ReLU) and this is explained by the modeling capacity of the former. Indeed, leaky ReLU reflects better the (positive and negative) values of our laplacians while ReLU cuts off all the negative values.

$L \gg 1$). This gain results from the *complementary aspects of the used elementary laplacians* and also the *match* between the topological properties of the learned multiple laplacians and the actual topology of the manifolds enclosing the input graphs. Besides, "expansion+GP" aggregates the representations of the learned GCN filters in a way that maintains their high discrimination power (at the node level) while achieving permutation invariance. The latter is clearly necessary especially when handing videos with multiple interacting persons that frequently appear in interchangeable orders (as in SBU and UCF).

Finally, we compare the performances of our MLGCN against related methods ranging from standard machine learning ones (SVMs [61, 88], sequence based such as LSTM and GRU [50, 51, 53], 2D/3D CNNs [16, 17, 18, 61] including appearance and motion streams) to deep graph (no-vectorial) methods based on spatial and spectral convolution [1, 2, 37]. From the results in table 4, MLGCN brings a substantial gain w.r.t state of the art graph-based methods on both sets, and provides comparable results with the best vectorial methods on SBU. On UCF, while vectorial methods are highly effective, their combination with our MLGCN (through a late fusion) brings an extra gain despite the fact that bridging the – last few percentage – gap is challenging, and this clearly shows its complementary aspect.

Pooling	Single-lapl		Multi-lap	
	SBU	UCF	SBU	UCF
No pooling	93.94	59.16	95.70	61.20
Global Pooling (GP)	93.90	59.10	95.62	61.17
Features prop [87]	94.27	59.30	96.36	61.31
Features prop [87] + GP	94.30	59.26	96.43	61.25
Exp ($r = 1, L = 1$)+GP	94.15	59.20	96.35	61.25
Exp ($r = 2, L = 1$)+GP	94.32	59.33	96.42	61.30
Exp ($r = 1, L = n$)+GP	**96.00**	**60.54**	**98.60**	**63.27**

(a)

Depth	Leaky ReLU		ReLU	
	SBU	UCF	SBU	UCF
1	**98.60**	63.10	98.57	63.07
2	98.56	**63.27**	98.52	63.25
3	98.30	63.27	98.23	63.23

(b)

skeleton representation	accuracy
Cloud of joints	31.65/34.25
Spatio-temporel skeletons	36.10/38.00
Orthocentred joints	43.25/45.80
Cylindrical features [12, 13]	38.42/40.10
3D coord +velocity features [39]	38.50/40.20
Joint joint orientation [42]	74.95/ 76.20
Joint line distance [42]	85.60/ 87.50
Our temporal chunking (sec 2)	**96.00/ 98.60**

(c)

Table 3: (a) Behavior of our MLGCN with and without expansion, i.e., after its ablation and replacement with other pooling methods. Note that results with the best single laplacians taken from tables 1 and 2 are also shown. (b) Behavior of our MLGCN w.r.t different depths and activation functions. (c) Performance of MLGCN on SBU for different state of the art skeleton graph/node representations; again results are also shown for the best underlying single laplacians (taken from tables 1 and 2). In this table, "Cloud of joints" stand for graphs based on the similarity between all the keypoints of different frames; "Spatio-temporel skeleton" graphs are obtained by computing intra-frame joint similarity and by connecting them to their predecessors and successors through frames; "Orthocentered joints" are obtained by centering the keypoint coordinates of each skeleton in each frame. Details about the other used node features (namely "Cylindrical features", "3D coord + velocity features", " Joint joint orientation" and "Joint line distance") can be found in [39, 40, 41, 42, 43, 44].

SBU		UCF	
Graph methods		Graph methods	
GCNConv [illegible]	90.00	GCNConv [illegible]	59.39
ArmaConv [illegible]	96.00	ArmaConv [illegible]	61.11
SGCCConv [illegible]	94.00	SGCCConv [illegible]	62.81
ChebyNet [illegible]	96.00	ChebyNet [illegible]	60.54
Our best MLGCN setting	**98.60**	Our best MLGCN setting	**63.27**
Vectorial methods		Vectorial methods	
Raw coordinates [illegible]	49.7	Temporal pyramid [illegible]	68.58
Joint features [illegible]	80.3	Potion [illegible]	64.38
Interact Pose [illegible]	86.9	Temporal pyramid [illegible] + our best MLGCN setting	70.37
CHARM [illegible]	83.9	Potion [illegible] + our best MLGCN setting	68.05
HBRNN-L [illegible]	80.35	Temporal pyramid [illegible] + Potion [illegible]	77.34
Co-occurrence LSTM [illegible]	90.41	Temporal pyramid [illegible] + Potion [illegible]+ our best MLGCN	79.10
ST-LSTM [illegible]	93.3	2D two stream [illegible]	91.12
Joint line distance [illegible]	**99.02**	2D two stream [illegible]+Our best MLGCN setting	93.20
Topological pose ordering [illegible]	90.5	3D appearance [illegible]	95.60
STA-LSTM [illegible]	91.51	3D appearance [illegible]+Our best MLGCN setting	95.92
GCA-LSTM [illegible]	94.9	3D motion [illegible]	96.41
VA-LSTM [illegible]	97.2	3D motion [illegible]+ our best MLGCN setting	96.60
DeepGRU [illegible]	95.7	3D two stream [illegible]	**97.94**
Riemannian manifold traj [illegible]	93.7		

Table 4: Comparison against state of the art methods.

5 Conclusion

We introduced in this paper a novel Multi-Laplacian Graph Convolutional Network (MLGCN) for action recognition. The strength of our method resides in its effectiveness in learning combined laplacian convolutional operators each one dedicated to a particular setting of the manifold enclosing the input graph data. Our method also considers a novel pooling process which first expands nodes with their context prior to achieve global average pooling. Extensive experiments conducted on the SBU as well as the challenging UCF-101 datasets, show the outperformance and also the complementary aspect of our MLGCN w.r.t different baselines and the related work including graph-based methods.
As a future work, we are currently studying other laplacian combination strategies and also the extension of our graph convolutional networks to other tasks and benchmarks.

Appendix

[Proof of Proposition 2]

Details of the first part of the proof, based on recursion, are omitted and result from the application of definition (1) to $\mathbf{L} = \sum_q \mathbf{w}^{\ell}_{q,p}\, \mathbf{L}^{\ell}_q$ (for different values of ℓ) while considering $\{\mathbf{L}^1_q\}_q$ c.p.d. Now we show the second part of the proof (i.e., if $\mathbf{L}$ is c.p.d, then $g(\mathbf{L})$ is also c.p.d for ReLU and leaky ReLU).

i) For $g(\mathbf{L}) = \log(1+\exp(\mathbf{L}))$ [ReLU]: considering $\mathbf{L}$ c.p.d, and following proposition (1), one may define a positive definite $\hat{\mathbf{L}}$ and obtain $\forall\{c_i\}$

$$\sum_{i,j=1}^{n} c_i c_j \exp(\mathbf{L}_{i,j}) \quad = \quad \exp(\mathbf{L}_{n+1,n+1}) \sum_{i,j=1}^{n} (c_i \exp(\mathbf{L}_{i,n+1})).(c_j \exp(\mathbf{L}_{n+1,j})).\exp(\hat{\mathbf{L}}_{i,j}) \geq 0$$

so $\exp(\mathbf{L})$ is also positive definite. Besides, for any arbitrary $\alpha > 0$, $(1+\exp(\mathbf{L}))^{\circ\alpha}$ is also positive definite with $\circ\alpha$ being the entry-wise matrix power. By simply rewriting $(1+\exp(\mathbf{L}))^{\circ\alpha} = \exp(\alpha\, g(\mathbf{L}))$, it follows (from [illegible]) that $g(\mathbf{L})$ is c.p.d since $\exp(\alpha\, g(\mathbf{L}))$ is positive definite for all $\alpha > 0$.

ii) For $g(\mathbf{L}) = \log(\exp(a\mathbf{L}) + \exp(\mathbf{L}))$ with $0 < a \ll 1$ [leaky-ReLU]: one may write g as

$$g(\mathbf{L}) = a\,\mathbf{L} + \log(1+\exp((1-a)\,\mathbf{L})). \tag{6}$$

Since $\exp(\mathbf{L})$ is positive definite, it follows that $(1+\exp((1-a)\,\mathbf{L}))^{\circ\alpha}$ is also positive definite for any arbitrary $\alpha > 0$ and $0 < a \ll 1$ so from [illegible], $\log(1+\exp((1-a)\,\mathbf{L}))$ is c.p.d and so is $g(\mathbf{L})$; the latter results from the closure of the c.p.d with respect to the sum.

■

References

[1] M. Defferrard, X. Bresson, P. Vandergheynst. Convolutional Neural Networks on graphs with Fast Localized Spectral Filtering. In Neural Information Processing Systems (NIPS), 2016

[2] TN. Kipf, M. Welling. Semi-supervised classification with graph convolutional networks. In International Conference on Learning Representations (ICLR), 2017

[3] H. Sahbi, X. Li. Context-based support vector machines for interconnected image annotation. Asian Conference on Computer Vision. Springer, Berlin, Heidelberg, 2010.

[4] M. Henaff, J. Bruna, Y. LeCun. Deep Convolutional Networks on Graph-Structured Data. In arXiv:1506.05163, 2015

[5] T. Postadjian, A. Le Bris, H. Sahbi, C. Mallet. Investigating the potential of deep neural networks for large-scale classification of very high resolution satellite images. ISPRS Annals 4, 183-190, 2017.

[6] J. Bruna, W. Zaremba, A. Szlam, Y. Lecun. Spectral Networks and Deep Locally Connected Networks on Graphs. In International Conference on Learning Representations (ICLR), 2014

[7] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, P. Vandergheynst. Geometric deep learning: going beyond Euclidean data. In IEEE Signal Processing Magazine, 2017

[8] D. Boscaini, J. Masci, E. Rodolà, M. M. Bronstein, Learning shape correspondence with anisotropic convolutional neural networks. In Neural Information Processing Systems (NIPS), 2016

[9] F. Yuan, G-S. Xia, H. Sahbi, V. Prinet. Mid-level Features and Spatio-Temporal Context for Activity Recognition. Pattern Recognition. volume 45, number 12, 4182-4191, 2012

[10] F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, and M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In computer Vision and Pattern Recognition (CVPR), 2017

[11] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Li,and Y. Bengio. Graph attention networks. In International Conference on Learning Representations (ICLR), 2018

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

[13] A. Ortega, P. Frossard, J. Kovacevic, Jose M. F. Moura, P. Vandergheynst. Graph Signal Processing: Overview, Challenges, and Applications. In Proceedings of the IEEE (Volume: 106 , Issue: 5 , May 2018) Page(s): 808 - 828.

[14] X. Li, H. Sahbi. Superpixel-based object class segmentation using conditional random fields. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2011.

[15] Z. Cao, T. Simon, S-E. Wei, Y. Sheikh. Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. In computer Vision and Pattern Recognition (CVPR), 2017

[16] V. Choutas, P. Weinzaepfel, J. Revaud, C. Schmid. PoTion: Pose MoTion Representation for Action Recognition. In computer Vision and Pattern Recognition (CVPR), 2018

[17] K. Simonyan, A. Zisserman. Two-Stream Convolutional Networks for Action Recognition in Videos. In Neural Information Processing Systems (NIPS), 2014

[18] J. Carreira, A. Zisserman. Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. In Computer Vision and Pattern Recognition (CVPR), 2017

[19] H. Pirsiavash, D. Ramanan. Detecting Activities of Daily Living in First-person Camera Views. In Computer Vision and Pattern Recognition (CVPR), 2012

[20] L. Chen, L. Duan, and D. Xu. Event Recognition in Videos by Learning From Heterogeneous Web Sources. In Computer Vision and Pattern Recognition (CVPR), 2013

[21] D. Xu, S-F. Chang. Visual Event Recognition in News Video using Kernel Methods with Multi-Level Temporal Alignment. In Computer Vision and Pattern Recognition (CVPR), 2013 2007

[22] H. Wang, C. Yuan, W. Hu, and C. Sun. Supervised class-specific dictionary learning for sparse modeling in action recognition. Pattern Recognition , 2012

[23] C. Schuldt, I. Laptev, B. Caputo. Recognizing human actions: a local SVM approach. In International Conference on Pattern Recognition Systems (ICPR), 2004

[24] C. Feichtenhofer, A. Pinz, R-P. Wildes. Spatiotemporal Residual Networks for Video Action Recognition. In Neural Information Processing Systems (NIPS), 2016

[25] C. Feichtenhofer, A. Pinz, R-P. Wildes. Spatiotemporal Multiplier Networks for Video Action Recognition. In Computer Vision and Pattern Recognition (CVPR), 2017

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

[27] S. Yan and Y. Xiong and D. Lin. Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. In Association for the Advancement of Artificial Intelligence (AAAI), 2018

[28] M. Schlichtkrull, T-N. Kipf, P. Bloem, R. Van den Berg, I. Titov, M. Welling. Modeling relational data with graph convolutional networks. In International Conference on Machine Learning (ICML), 2019

[29] R. Van den Berg, T-N. Kipf, M. Welling. Graph Convolutional Matrix Completion. In arXiv:1706.02263, 2017

[30] I. Dhillon, Y. Guan, and B. Kulis. Weighted Graph Cuts Without Eigenvectors: A Multilevel Approach. In IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 29(11):1944–1957, 2007

[31] M. Jiu, H. Sahbi. Nonlinear deep kernel learning for image annotation. IEEE Transactions on Image Processing, volume 26, number 4, 1820-1832, 2017.

[32] WL. Hamilton, R. Ying, J. Leskovec. Inductive Representation Learning on Large Graphs. In Neural Information Processing Systems (NIPS), 2018

[33] C. Morris, M. Ritzert, M. Fey, WL. Hamilton, JE. Lenssen, G. Rattan, M. Grohe. Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks. In arXiv:1810.02244, 2018

[34] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio. Graph Attention Networks. In International Conference on Learning Representation (ICLR), 2018

[35] K. Thekumparampil, C. Wang, S. Oh, L-J. Li. Attention-based Graph Neural Network for Semi-supervised Learning. In arXiv:1803.03735, 2018

[36] F-M. Bianchi, D. Grattarola, C. Alippi, L. Livi. Graph Neural Networks with Convolutional ARMA Filters. In arXiv:1901.01343, 2019

[37] F. Wu, T. Zhang, A. Holanda de Souza Jr., C. Fifty, T. Yu, K-Q. Weinberger. Simplifying Graph Convolutional Networks. In arXiv:1902.07153, 2019

[38] J. Klicpera, A. Bojchevski, S. Günnemann. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In International Conference on Learning Representation (ICLR), 2019

[39] M. Zanfir, M. Leordeanu, and C. Sminchisescu. The moving pose: An efficient 3d kinematics descriptor for low-latency action recognition and detection. In International Conference on Computer Vision (ICCV), 2013

[40] F. Baradel, C. Wolf, J. Mille. Pose-conditioned Spatio-Temporal Attention for Human Action Recognition. In arXiv preprint, 2017

[41] D. Weinland, R. Ronfard, E. Boyer. Free viewpoint action recognition using motion history volumes. In Computer vision and image understanding 104 (2-3), 249-257

[42] Q. Ke, M. Bennamoun, S. An, F. Sohel, F. Boussaid. A New Representation of Skeleton Sequences for 3D Action Recognition. In Computer Vision and Pattern Recognition (CVPR), 2017

[43] Beyer, W. H. CRC Standard Mathematical Tables, 28th ed. Boca Raton, FL: CRC Press, 1987

[44] S. Zhang, X. Liu, J. Xiao. On Geometric Features for Skeleton-Based Action Recognition using Multilayer LSTM Networks. In Conference on Applications of Computer Vision (WACV), 2017

[45] Y. Ji, G. Ye, and H. Cheng. Interactive body part contrast mining for human interaction recognition. In International Conference on Multimedia and Expo Workshops (ICMEW), 2014

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

[47] Y. Du, W. Wang, and L. Wang. Hierarchical recurrent neural network for skeleton based action recognition. In Computer Vision and Pattern Recognition (CVPR), 2015

[48] 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 Association for the Advancement of Artificial Intelligence (AAAI), 2016

[49] 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 (ECCV), 2016

[50] 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 Association for the Advancement of Artificial Intelligence (AAAI), 2017

[51] 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):1586–1599, April 2018

[52] 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. In International Conference on Computer Vision (ICCV), 2017

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

[54] L. Wang, H. Sahbi. Nonlinear Cross-View Sample Enrichment for Action Recognition. European Conference on Computer Vision. Springer, 2014.

[55] 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

[56] M. Fey, J-E. Lenssen, F. Weichert, H. Müller. SplineCNN: Fast geometric deep learning with continuous B-spline kernels. In Computer Vision and Pattern Recognition (CVPR), 2018

[57] L. Wang, H. Sahbi. Directed Acyclic Graph Kernels for Action Recognition. Proceedings of the IEEE International Conference on Computer Vision. 2013.

[58] L. Shi, Y. Zhang, J. Cheng, H. Lu. Adaptive Spectral Graph Convolutional Networks for Skeleton-Based Action Recognition. In arXiv preprint arXiv:1805.07694, 2018

[59] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In European Conference on Computer Vision (ECCV), 2016

[60] K. Thakkar, P-J. Narayanan. Part-based Graph Convolutional Network for Action Recognition. In British Machine Vision Conference (BMVC), 2018

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

[62] Z. Wu and al. A Comprehensive Survey on Graph Neural Networks. In arXiv:1901.00596, 2019

[63] B. Xu, H. Shen, Q. Cao, Y. Qiu, X. Cheng. Graph Wavelet Neural Network. In International Conference of Learning Representation (ICLR), 2019

[64] D. Zou, G. Lerman. Graph Convolutional Neural Networks via Scattering. In arXiv preprint arXiv:1804.00099, 2018

[65] F. Gama, A. Ribeiro, J. Bruna. Diffusion Scattering Transforms on Graphs. In International Conference of Learning Representation (ICLR), 2019

[66] S. Thiemert, H. Sahbi, M. Steinebach. Applying interest operators in semi-fragile video watermarking. Security, Steganography, and Watermarking of Multimedia Contents VII. Vol. 5681. International Society for Optics and Photonics, 2005.

[67] M. Zhang, Z. Cui, M. Neumann, Y. Chen. An End-to-End Deep Learning Architecture for Graph Classification. In Association for the Advancement of Artificial Intelligence (AAAI), 2018

[68] O. Vinyals, S. Bengio, M. Kudlur. Order Matters: Sequence to sequence for sets. In International Conference on Learning Representation (ICLR), 2016

[69] C. Cangea, P. Velickovic, N. Jovanovic, T. Kipf, P. Lio. Towards sparse hierarchical graph classifiers. In arXiv preprint arXiv:1811.01287, 2018

[70] M. Simonovsky, N. Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In Computer Vision and Pattern Recognition (CVPR), 2018

[71] CR. Qi, L. Yi, H. Su, LJ Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In Neural Information Processing Systems (NIPS), 2017

[72] R. Ying, J. You, C. Morris, X. Ren, W-L. Hamilton, J. Leskovec. Hierarchical Graph Representation Learning with Differentiable Pooling. In Neural Information Processing Systems (NIPS), 2018

[73] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In Computer Vision and Pattern Recognition (CVPR), 2016.

[74] P. Hermosilla, T. Ritschel, PP. Vazquez, A. Vinacua, T. Ropinski. Monte Carlo convolution for learning on non-uniformly sampled point clouds. In ACM SIGGRAPH, 2018

[75] B. Girault, A. Ortega, S. Narayanan. Irregularity-aware graph Fourier transforms. IEEE Transactions on Signal Processing 66 (21), 5746-5761

[76] DI. Shuman, B. Ricaud, P. Vandergheynst. Vertex-frequency analysis on graphs. Applied and Computational Harmonic Analysis 40 (2), 260-291, 2016

[77] R. Kondor, HT. Son, H. Pan, B. Anderson, S. Trivedi. Covariant compositional networks for learning graphs. In arXiv preprint arXiv:1801.02144, 2017

[78] X. Dong, D. Thanou, P. Frossard, P. Vandergheynst. Learning Laplacian matrix in smooth graph signal representations. IEEE Transactions on Signal Processing 64 (23), 6160-6173, 2016

[79] A. Loukas, P. Vandergheynst. Spectrally approximating large graphs with smaller graphs. In Internationa Conference on Machine Learning (ICML), 2018

[80] L. Wang, H. Sahbi. Bags-of-Daglets for Action Recognition. IEEE International Conference on Image Processing (ICIP), 2014.

[81] X. Glorot, A. Bordes and Y. Bengio. Deep sparse rectifier neural networks. In International Conference on Artificial Intelligence and Statistics (AISTATS), 2011

[82] C. Dugas, Y. Bengio, F. Belisle, C. Nadeau, R. Garcia. Incorporating second-order functional knowledge for better option pricing. In Neural Information Processing Systems (NIPS), 2000

[83] H. Kaiming, Z. Xiangyu, R. Shaoqing, S. Jian. Delving Deep into Rectifiers: Surpassing Human-Level Performance on Image Net Classification. arXiv:1502.01852, 2015.

[84] S. Tollari, P. Mulhem, M. Ferecatu, H. Glotin, M. Detyniecki, P. Gallinari, H. Sahbi, Z-Q. Zhao. A comparative study of diversity methods for hybrid text and image retrieval approaches. In Workshop of the Cross-Language Evaluation Forum for European Languages, pp. 585-592. Springer, Berlin, Heidelberg, 2008.

[85] Berg, Christian, Jens Peter Reus Christensen, and Paul Ressel. Harmonic analysis on semigroups: theory of positive definite and related functions. Vol. 100. New York: Springer, 1984

[86] Schoenberg, Isaac J. Metric spaces and positive definite functions. Transactions of the American Mathematical Society 44.3 (1938): 522-536

[87] A. Mazari, H. Sahbi. BMVC19 Supplementary Material. http://www-ia.lip6.fr/∼sahbi/SMBM19.pdf

[88] K. Yun and J. Honorio and D. Chattopadhyay and T-L. Berg and D. Samaras. Two-person Interaction Detection Using Body-Pose Features and Multiple Instance Learning. In Computer Vision and Pattern Recognition (CVPR), 2012

[89] K. Soomro, A-R. Zamir and M. Shah. UCF101: A Dataset of 101 Human Action Classes From Videos in The Wild. In Computer Vision and Pattern Recognition (CVPR), 2012.

[90] H. Sahbi, J-Y. Audibert, J. Rabarisoa, R. Keriven, R. Context-dependent kernel design for object matching and recognition. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2008.