Kernel-based Graph Convolutional Networks

Hichem Sahbi

▶ **To cite this version:**

Hichem Sahbi. Kernel-based Graph Convolutional Networks. ICPR 2020 - 25th International Conference on Pattern Recognition, Jan 2021, Milan / Virtuel, Italy. hal-03079770

HAL Id: hal-03079770
https://hal.science/hal-03079770

Submitted on 17 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.

Kernel-based Graph Convolutional Networks

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

***Abstract*—Learning graph convolutional networks (GCNs) is an emerging field which aims at generalizing deep learning to arbitrary non-regular domains. Most of the existing GCNs follow a neighborhood aggregation scheme, where the representation of a node is recursively obtained by aggregating its neighboring node representations using averaging or sorting operations. However, these operations are either ill-posed or weak to be discriminant or increase the number of training parameters and thereby the computational complexity and the risk of overfitting.**
In this paper, we introduce a novel GCN framework that achieves spatial graph convolution in a reproducing kernel Hilbert space. The latter makes it possible to design, via implicit kernel representations, convolutional graph filters in a high dimensional and more discriminating space without increasing the number of training parameters. The particularity of our GCN model also resides in its ability to achieve convolutions without explicitly realigning nodes in the receptive fields of the learned graph filters with those of the input graphs, thereby making convolutions permutation agnostic and well defined. Experiments conducted on the challenging task of skeleton-based action recognition show the superiority of the proposed method against different baselines as well as the related work.

***Index Terms*—graph convolutional networks, kernel machines, action recognition**

I. Introduction

There is an increasing interest in deep learning for different pattern classification and recognition tasks [1], [2], [13]. These parametric models rely on deep neural networks, composed of several convolutional, pooling and fully connected layers, that capture different levels of abstractions in the analyzed patterns [43]. These models have been popular in the analysis of vectorial data; i.e., those sitting on top of regular domains such as images [44], [45], [47]–[49]. However, the extension of these models to non-regular domains, such as graphs, remains a major challenge even though interesting solutions are currently emerging [4], [14], [15], [22], [24], [34]. Indeed, the difficulty in analyzing non-vectorial data stems from the ambiguity in defining usual operations namely convolutions. Whereas achieving convolution using sliding windows in regular domains, such as images, is a well defined operation, there is no clear definition of sliding windows in general graphs [34]; besides, the number and the order of nodes that intervene in the receptive fields of convolutions may change dramatically across different graph instances.

Early graph convolutional network (GCN) methods [18], [33], [38], [41] and their variants (see for instance [12], [29], [72], [74]–[77]) are rather spatial and seek to learn graph representations by iteratively propagating node features (a.k.a representations, descriptions or signals) through their neighbors using recurrent neural architectures till a stationary point is reached. These spatial methods also include recurrent gaited networks [12], [29], [38] that share the same convolutional parameters through layers, and composition-based convolutional networks [20] that consider different parameters. However, on highly irregular graphs, convolutions are ill-posed as the notion of translation and filter support (i.e., receptive field) cannot be consistently defined. Existing attempts, to address these issues, achieve node sorting and efficient sampling of neighboring nodes in order to define the receptive field during graph convolutions [6] and to make it similar to regular (grid-like) domains [3], [15], [35]. Other solutions operate differently [15], [20], [34], [35], [42]; first, they describe nodes by aggregating their neighbors into fixed length features prior to apply convolution (based on inner product) on the aggregated features.

On another hand, spectral methods provide interesting alternatives to make convolutions well defined [4], [14], [21], [24], [27], [28], [71]. These methods rely on the Fourier transform that projects the signal of a given graph using the spectral decomposition of its Laplacian prior to perform convolution in the Fourier domain, and then back-project the result in the input domain; in particular, the method in [14] makes it possible to project graph signals using an orthogonal Chebyshev basis prior to achieve convolution. An extension, in [24], allows to reduce the Chebychev polynomial using a first order approximation which provides a spatially localized convolution, that is equivalent to spatial methods. A variant in [8] interprets the graph convolutions in [24] as integral transforms of embedding functions under probability measures and uses Monte Carlo sampling to efficiently and consistently estimate the integrals. Huang et al. [22] propose an adaptive layer-wise sampling approach, based on variance reduction in order to accelerate the training of ChebyshevNet [24], where sampling for a lower layer is conditioned on a top one. Nonetheless, most of these spectral methods suffer from several drawbacks; the eigen decomposition of the Laplacian, besides being computationally expensive, is sensitive to any small perturbation of input graphs (that may result from the intra-class variability). Moreover, the learned filters are domain dependent and cannot be transferred to graphs with high topological variations.

Besides the aforementioned issues, the accuracy of both spatial and spectral GCNs also relies on the discrimination power of the input graph signal. For highly nonlinear graph signals, relying on convolutions in the input space may limit the discrimination power of the learned convolutional representations

and may result into limited accuracy. Furthermore, sorting using automorphisms is not always consistent through different graph instances while aggregation based on averaging (when achieved in the input space) may dilute input node representations prior to convolution. An explicit expansion of the input node representations may enhance the discrimination power but comes at the expense of a substantial increase in the number of training parameters (thereby the risk of overfitting) and also an increase in the computational complexity both in space and time. Therefore, one should consider, instead, an *implicit* mapping of the input graph signal in a (high or possibly infinite dimensional) reproducing kernel Hilbert space (*RKHS*) [81] and achieve an averaging aggregation and convolution in that space, in order to enhance the representational power of nodes and also the learned graph representations while being permutation agnostic. This mapping scheme has been proven to be effective in kernel methods, and particularly in support vector machines (SVMs) (see for instance [7], [25], [50], [51], [60], [65], [73]) and it is extended, in our paper, to GCNs.

Considering these challenges, we introduce in this paper a *dual* formulation of GCN based on kernels which maps graph signals from an input space into a high dimensional Hilbert space. This mapping is implicitly defined using positive semi-definite kernels that enhance the discrimination power of the learned graph representations, without explicitly increasing the dimensionality of the input graph signals nor the number of training parameters[1]. This is beneficial when handling low dimensional raw signals, such as 3D skeleton graphs in action recognition [52], [53]; indeed the low dimensionality of these data makes the Bayes risk of the underlying classification task intrinsically high, and this requires increasing the dimensionality of the input raw signal. Moreover, our GCN achieves convolutions without explicitly realigning nodes in the receptive fields of the learned graph filters with those of the input graphs, thereby making convolutions permutation agnostic. We cast the problem of filter design as kernel learning with the particularity of using standard kernels while training only their support vectors; this scheme of learning the support vectors (as a part of kernel design) is conceptually different from the two major families of kernel learning techniques, namely non-parametric [40] and parametric ones [10], [17][2]. Finally, extensive experiments on the challenging task of action recognition show the high gain of our kernel-based GCNs w.r.t standard baselines as well as the related work.

II. Graph convolutional networks

Let $\mathcal{S} = \{\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)\}_i$ denote a collection of graphs with $\mathcal{V}_i$, $\mathcal{E}_i$ being respectively the nodes and the edges of $\mathcal{G}_i$. Each graph $\mathcal{G}_i$ (denoted for short as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$) is endowed with (i) a signal $\{s(u) \in \mathcal{X} :\ u \in \mathcal{V}\}$ (with $\mathcal{X} = \mathbb{R}^D$ being an input space) and (ii) a row-stochastic adjacency matrix $\mathbf{A}$ with each entry $\mathbf{A}_{uu'} > 0$ iff $(u, u') \in \mathcal{E}$ and 0 otherwise. Our goal is to design a novel graph convolutional network that returns both the representation and the classification of $\mathcal{G}$.

A. Standard graph convolutional networks

Consider $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $g_\theta = (\mathcal{V}_\theta, \mathcal{G}_\theta)$ as two graphs with $|\mathcal{V}_\theta| \ll |\mathcal{V}|$ and $|\mathcal{E}_\theta| \ll |\mathcal{E}|$. Following standard GCNs (see for instance [34]), the spatial convolution of $\mathcal{G}$ with a graph g_θ at a given node $u \in \mathcal{V}$ is defined as

$$(\mathcal{G} \star g_\theta)_u = \sigma(\mathcal{K}_\theta(u)), \tag{1}$$

with

$$\mathcal{K}_\theta(u) = \left\langle \sum_{u'} s(u').[\mathbf{A}^r]_{uu'}, w_\theta \right\rangle, \tag{2}$$

here $\sigma(.)$ is a nonlinear activation (taken in practice as ReLU), $w_\theta \in \mathcal{X}$ corresponds to the filter parameters of the graph g_θ (also referred to as graphlet) and $[\mathbf{A}^r]_{uu'}$ is the u'^{th} column of the u^{th} row of the r-hop adjacency matrix $\mathbf{A}^r$. In this definition, the left-hand side term of the inner product in Eq. 2, aggregates the neighbors of u into a single vector prior to multiply this vector by w_θ.

In spite of being agnostic to any arbitrary permutation of nodes in $\mathcal{G}$, the above definition suffers from limited discrimination power, as the signal informations in the neighborhood system $\{\mathcal{N}_r(u)\}_u$ of $\mathcal{G}$ are mixed during convolution. In what follows, we consider a dual convolutional operator, based on kernels, that overcomes this limitation and provides more discriminating convolutional representations while still being agnostic to any arbitrary permutation of nodes in graphs.

B. Our kernel-based graph convolutional networks

Considering κ as a symmetric positive definite function (i.e., $\exists \psi : \mathcal{X} \to \mathcal{H}$, with ψ being an implicit mapping that takes graph signals from an input space $\mathcal{X}$ to a high dimensional Hilbert space $\mathcal{H}$, s.t., $\kappa(s(u'), s(v)) = \langle \psi(s(u')), \psi(s(v)) \rangle$) and for a particular setting of w_θ as $\frac{1}{|\mathcal{V}_\theta|} \sum_{i=1}^{N} \alpha_i^\theta \psi(s(v_i^\theta))$[3], with $\{v_i^\theta\}_i \subset \mathcal{V}_\theta$, $\{\alpha_i^\theta\}_i \subset \mathbb{R}$; the convolutional operator defined in Eq. 2 can be rewritten as

$$\mathcal{K}_\theta(u) = \frac{1}{|\mathcal{N}_r(u)|.|\mathcal{V}_\theta|} \sum_{u' \in \mathcal{N}_r(u)} \left(\sum_{i=1}^{N} \alpha_i^\theta \ \kappa(u', v_i^\theta) \right), \tag{3}$$

here $\mathcal{N}_r(u)$ is the set of r-hop neighbors of u and $\kappa(s(.), s(.))$, $\psi(s(.))$ are written for short as $\kappa(.,.)$ and $\psi(.)$ respectively. In the above definition, $\{v_i^\theta\}_{i,\theta}$ are referred to as support vectors and $\alpha = \{\alpha_i^\theta\}_{i,\theta}$ as the underlying mixing parameters. Since $\mathcal{K}_\theta(u)$ is defined as the sum of all of the kernel values between all of the possible signal pairs taken from $\mathcal{N}_r(u) \times \mathcal{V}_\theta$, its evaluation does not require any explicit alignment between

[1] In contrast to [11], [19], [23], [26], [37], [39] which may increase the number of parameters in the model and the risk of overfitting.

[2] In "non-parametric" training, the number of parameters follows exactly the size of training data (e.g., nonlinear SVMs) while in "parametric" training, this number is fixed independently (e.g., linear SVMs). In "semi-parametric" training, only a fraction of the parameters follows proportionally the size of training data.

[3] This setting is related to the representer theorem widely used in kernel methods [79], [80]. The latter states that many problems have optimal solutions that live in a finite dimensional span of training data mapped into a high dimensional Hilbert space, and this makes it possible to define kernel-based algorithms independently of the (high or infinite) dimensionality of these Hilbert spaces.

these pairs and it is thereby still invariant to any arbitrary permutation (including rotations) of nodes in $\mathcal{V}$ and $\mathcal{V}_\theta$.

The strength of this kernel trick resides in its capacity to handle nonlinear data as node representations are mapped into a high dimensional (and more discriminating) space $\mathcal{H} = \mathbb{R}^H$. For instance, when using the polynomial kernel $\kappa(s(u), s(v)) = \langle s(u), s(v)\rangle^p$, its underlying mapping is explicitly defined as $\psi(s(u)) = s(u) \otimes \cdots \otimes s(u)$ (with $\otimes$ being the Kronecker tensor product applied $p-1$ times); see also [31], [32], [39]. As the dimensionality H of this explicit map grows exponentially w.r.t p and polynomially w.r.t D, the kernel form is rather computationally more efficient. Indeed, considering a non-parametric setting with a fixed set of support vectors $\{v_i^\theta\}_i$ taken from the training set (i.e., $\cup\, \mathcal{V}_j$); when only $\{\alpha_i^\theta\}_i$ are allowed to vary in $w_\theta = \frac{1}{|\mathcal{V}_\theta|}\sum_i \alpha_i^\theta \psi(v_i^\theta)$, and when $H \gg |\{\alpha_i^\theta\}_i|$, the kernel trick presented earlier provides a computational and a generalization advantage (i.e., the convolution in Eq. 3 has fewer parameters compared to the one in Eq. 2). However, this may still come at the expense of a *quadratic* complexity when naively evaluating $\{\kappa(.,.)\}$; for mid (and even small) scale training problems with a large number of nodes in $\cup\, \mathcal{V}_j$, this complexity becomes clearly intractable.

One question that arises is how to make this approach parametric (or at least semi-parametric); in other words, how to maintain the kernel trick advantage (in Eq. 3) without significantly increasing the computational cost w.r.t the total number of nodes in $\cup\, \mathcal{V}_j$. Solutions such as sampling and reduced set technique [5] are both limited; on the one hand, sampling may generate a smaller fixed set of support vectors $\{v_i^\theta\}_i$ but biased (i.e., very limited to comprehensively make w_θ a universal filter approximator). On the other hand, the reduced set technique requires first building an initial expensive model w_θ before reducing its complexity by solving a difficult pre-image optimization problem [5], [46]. Our alternative, in this work, is to control the size of $\{v_i^\theta\}_i$ while allowing entries in $\{v_i^\theta\}_i$ to vary as a part of the end-to-end GCN (and also kernel) learning; this makes it possible to model a larger class of filters $\{w_\theta\}$ that better fit the classification task at hand (see later experiments).

Note that one may consider a kernel approximation $\hat{\psi}(.)$ s.t. $\kappa(.,.) \approx \langle \hat{\psi}(.), \hat{\psi}(.)\rangle$ (as done in [9], [11], [19], [23], [26], [30], [36], [37], [39] which seek to handcraft or learn shallow/deep explicit maps whose inner products approximate the original kernel values) and use instead Eq. 2. However, this approximation usually results into very high dimensional mappings (and hence into a lot of training parameters in w_θ), especially when considering highly nonlinear (and also discriminative) kernels such as gaussian and histogram intersection. Put differently, even when learning both $\{v_i^\theta\}_i$ and $\{\alpha_i^\theta\}_i$, the dual formulation in Eq. 3 is computationally more efficient and less subject to overfitting, as the dimensionality H of $\hat{\psi}$ is often $\gg |\mathcal{V}_\theta| \times D$ for the widely used kernels including gaussian and histogram intersection. In sum, our method is rather targeted to learn kernels following a (semi-)parametric setting by allowing the support vectors of these kernels to be learned (instead of being taken from training data) and this is also conceptually very different from multiple kernel learning [17].

C. Neural consistency and architecture design

In contrast to usual convolutional operators on graphs (including Eq. 2), the one in Eq. 3 cannot be straightforwardly evaluated using standard neural units[4] as kernels may have general forms. Hence, modeling Eq. 3 requires a careful design; our goal in this paper, is not to change the definition of neural units, but instead to adapt Eq. 3 in order to make it consistent with the usual definition of neural units. In what follows, we introduce the overall architecture associated to $\mathcal{K}_\theta(.)$ (and the whole GCN) for different kernels including linear, polynomial, gaussian and histogram intersection as well as a more general class of shift invariant kernels.

Definition 1 (Neural consistency): Let $u_{.,d}$ (resp. $v_{.,d}$) denote the d^{th} dimension of the signal in a given node u (resp. v). For a given (fixed or learned) v, a kernel κ is referred to as "neural-consistent" if

$$\kappa(u, v) = \sigma_3\Big(\sum_d \sigma_2(\sigma_1(u_{.,d}).\omega_d)\Big), \tag{4}$$

with $\omega_d = \sigma_4(v_{.,d})$ and being σ_1, σ_2, σ_3, σ_4 any arbitrary real-valued activation functions.

Considering the above definition, the following kernels are neural consistent: linear $\langle u, v\rangle$, polynomial $\langle u, v\rangle^p$, and $\tanh(a\langle u, v\rangle + b)$. Neural consistency is straightforward for inner product-based kernels (namely linear, polynomial and tanh) while for shift-invariant ones such as the gaussian, one may obtain neural consistency by rewriting $\exp(-\beta\|u-v\|_2^2) = \sigma_3\big(\sum_d \sigma_2(\sigma_1(u_{.,d}).\omega_d)\big)$ with $\sigma_1(.) = \exp(.)$, $\sigma_2(.) = \log(.)^2$, $\sigma_3(.) = \exp(-\beta(.))$ and $\omega_d = \exp(-v_{.,d})$. Other kernels (including Laplacian, inverse multiquadric, power, log, Cauchy[5]) are also neural consistent (see table I for the setting of their σ_1, σ_2, σ_3, σ_4).

For the histogram intersection kernel, $\sum_d \min(u_{.,d}, v_{.,d}) = \sum_d 1 - \max(1 - u_{.,d}, 1 - v_{.,d})$ and one may easily obtain $\sum_d 1 - \max(1-u_{.,d}, 1-v_{.,d}) \approx \sigma_3\big(\sum_d \sigma_2(\sigma_1(u_{.,d}).\omega_d)\big)$ using $\sigma_1(.) = \exp(\exp(\beta(1-(.))))$, $\sigma_2(.) = -\frac{1}{\beta}\log(\log(.)) + 1$, $\sigma_3(.) = (.)$ and $\omega_d = \sigma_1(v_{.,d})$ (for a sufficiently large β). In the following section, we discuss the implementation details of our global GCN architecture built on top of these neural consistent kernels.

Implementation: Fig. 1 shows the architecture of our deep net including kernel evaluation and the weighted convolution blocks. The former block is fed with the input graph signal $s(\mathcal{V})$ (denoted for short as $\mathcal{V}$) and the adjacency matrix $\mathbf{A}$ following the same arbitrary order both in $\mathcal{V}$ and $\mathbf{A}$. In the first layer, the σ_1 activation is first applied to all the dimensions of the signal $\mathcal{V}$, then each dimension of the resulting activated

[4] i.e., those based on standard perceptron (inner product operators) followed by nonlinear activations.

[5] See for instance [16] for a taxonomy of the widely used functions in kernel machines.

		$\kappa(u,v)$	$\sigma_1(t)$	$\sigma_2(t)$	$\sigma_3(t)$	$\sigma_4(t)$
Inner product based	Linear	$\langle u,v\rangle$	t	t	t	t
	Polynomial	$\langle u,v\rangle^p$	t	t	t^p	t
	Sigmoid	$\frac{1}{1+\exp(-\beta\langle u,v\rangle)}$	t	t	$\frac{1}{1+\exp(-\beta t)}$	t
	tanh	$\tanh(a\langle u,v\rangle+b)$	t	t	$\tanh(at+b)$	t
Distance based	Gaussian	$\exp(-\beta\|u-v\|^2)$	$\exp(t)$	$\log(t)^2$	$\exp(-\beta t)$	$\exp(-t)$
	Laplacian	$\exp(-\beta\|u-v\|)$	$\exp(t)$	$\log(t)^2$	$\exp(-\beta\sqrt{t})$	$\exp(-t)$
	Power	$-\|u-v\|^p$	$\exp(t)$	$\log(t)^2$	$-t^{p/2}$	$\exp(-t)$
	Inverse Multi-quadric	$\frac{1}{\sqrt{\|u-v\|^2+b^2}}$	$\exp(t)$	$\log(t)^2$	$\frac{1}{\sqrt{t+b^2}}$	$\exp(-t)$
	Log	$-\log(\|u-v\|^p+1)$	$\exp(t)$	$\log(t)^2$	$-\log(t^{p/2}+1)$	$\exp(-t)$
	Cauchy	$\frac{1}{1+\frac{\|u-v\|^2}{\sigma^2}}$	$\exp(t)$	$\log(t)^2$	$\frac{1}{1+\frac{t}{\sigma^2}}$	$\exp(-t)$
	Histogram intersection	$\sum_d \min(u_{.,d}, v_{.,d})$	$\exp(\exp(\beta(1-t)))$	$-\frac{1}{\beta}\log(\log(t))+1$	t	$\sigma_1(t)$

TABLE I
THIS TABLE SHOWS THE SETTING OF σ_1, σ_2, σ_3, σ_4 FOR DIFFERENT KERNEL FUNCTIONS.

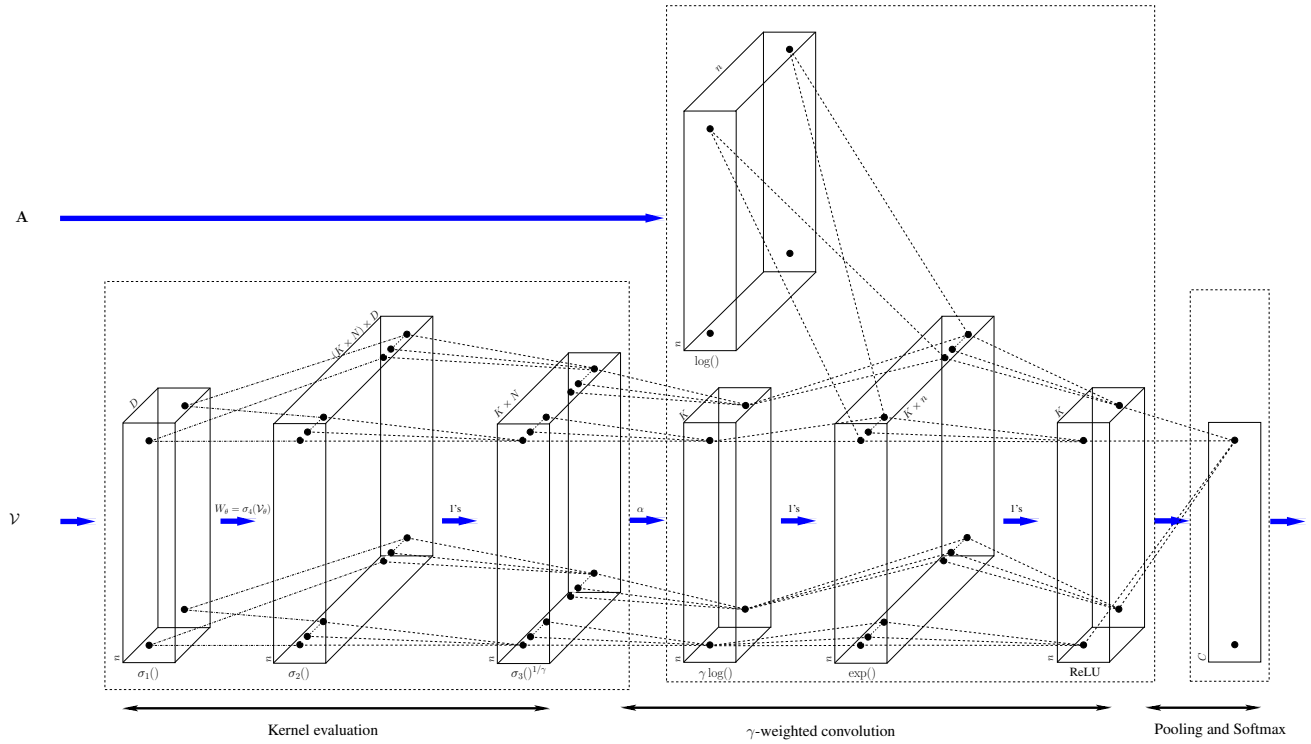

Fig. 1. This figure shows the architecture of our kernel-based graph convolutional network; see a detailed description of this architecture in the "Implementation Part" of section II-C **(better to zoom the PDF version).**

signal $\sigma_1(\mathcal{V})$ is multiplied, in the second layer, by the $K \times N$ (reparameterized) weights of the node filters $\{\sigma_4(\mathcal{V}_\theta)\}_\theta$ (as shown in Eqs. 3, 4) prior to apply the σ_2 activation; here K corresponds to the number of filters and N the number of nodes in the expansion of each filter. Note that these weights are shared through different nodes in $\mathcal{V}$. In the third layer, the results of the previous one are pooled across dimensions resulting into $N \times K$ kernel values per node in $\mathcal{V}$. These kernel values are activated by $\sigma_3()$ and fed to the weighted convolutional block in order to evaluate their weighted linear combinations, in the fourth layer, resulting into K pooled kernel values per node (see again Eqs. 3, 4). These pooled kernel values are crossed, in the fifth layer, with the nonzero entries of the adjacency matrix $\mathbf{A}$ in order to make the receptive field of the convolutional operation local. Note also that the activation functions $\log()$ and $\exp()$ are successively applied in the fourth and fifth layers in order to make this crossing operation neural consistent. Indeed, one may rewrite Eq. 3 as

$$\mathcal{K}_\theta(u) \quad = \quad \frac{1}{|\mathcal{V}_\theta|}\sum_{u'} \exp\left(\log \mathbf{A}_{uu'} + \log \sum_{i=1}^{N} \alpha_i^\theta \ \kappa(u', v_i^\theta)\right), \tag{5}$$

which corresponds to the neural consistent form shown in Eq. 4. The results of this fifth layer are pooled, in the sixth layer, through the neighborhood systems $\{\mathcal{N}_r(u)\}_u$ and fed to the ReLU activation resulting into K features per node in $\mathcal{V}$. Finally, these node features are used for final pooling and

softmax classification.

III. Experimental validation

We evaluate the performance of our kernel-based GCN (KGCN) on the challenging task of action recognition, using the SBU kinect dataset [53]. The latter is an interaction dataset acquired using the Microsoft kinect sensor; it includes in total 282 video sequences belonging to $C = 8$ categories: "approaching", "departing", "pushing", "kicking", "punching", "exchanging objects", "hugging", and "hand shaking" with variable duration, viewpoint changes and interacting individuals (see examples in Fig. 2). In all these experiments, we use the same evaluation protocol as the one suggested in [53] (i.e., train-test split) and we report the average accuracy over all the classes of actions.

A. Video skeleton description

Given a video $\mathcal{V}$ in SBU as a sequence of skeletons, each keypoint in these skeletons defines a labeled trajectory through successive frames (see Fig. 2). Considering a finite collection of trajectories $\{v_j\}_j$ in $\mathcal{V}$, we process each trajectory using *temporal chunking*: first we split the total duration of a video into M equally-sized temporal chunks ($M = 8$ in practice), then we assign the keypoint coordinates of a given trajectory v_j to the M chunks (depending on their time stamps) prior to concatenate the averages of these chunks and this produces the description of v_j (again denoted as $s(v_j) \in \mathbb{R}^D$ with $D = 3 \times M$) and $\{s(v_j)\}_j$ constitutes the raw description of nodes in a given video $\mathcal{V}$. Note that two trajectories v_j and v_k, with similar keypoint 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.

B. Performances and comparison

We trained our kernel-based GCN end-to-end for 3000 epochs with a batch size equal to 50, a momentum of 0.9 and we set the learning rate (denoted as ν) iteratively inversely proportional to the speed of change of the cross entropy loss used to train our network; when this speed increases (resp. decreases), ν decreases as $\nu \leftarrow \nu \times 0.99$ (resp. increases as $\nu \leftarrow \nu/0.99$). All these experiments are run on a GeForce GTX 1070 GPU device (with 8 GB memory) and no data augmentation is achieved. Table II shows a comparison of action recognition performances (and also runtime per epoch during training), using our KGCN (with different kernels) against standard GCN (referred to as SGCN), shown in section II-A, with precomputed node representations based on kernel principal component analysis (KPCA) achieved on

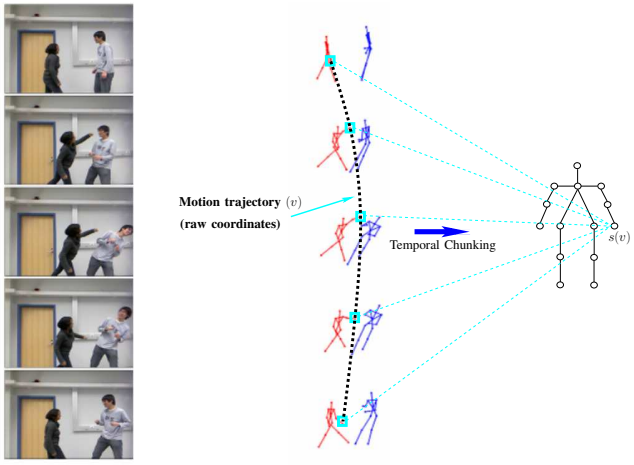

Fig. 2. This figure shows the whole keypoint tracking and description process.

$\{s(u) : u \in \cup \mathcal{V}_i\}$ using different kernels; in these results, we consider different numbers of eigenvectors (projection axes) corresponding to the largest eigenvalues of KPCA.

From all these results in table II, we observe a clear and a consistent gain of KGCN w.r.t the linear version (i.e., KGCN with linear kernel), as well as SGCN combined with different KPCA features; we observe an increase of the accuracy of the SGCN baseline when the dimension of KPCA (again denoted as H) is sufficiently large (without being able to overtake KGCN for most of the kernels) and performances decrease again as the underlying number of training parameters follows H and this may lead to overfitting. Besides, the average runtime per epoch, with SGCN, increases substantially when H grows, as the number of training parameters in the underlying network (equal to $H \times K + C \times K$) depends on H while in KGCN the number of training parameters (equal to $(D+1) \times N \times K + C \times K$) depends only on the dimension D of the original signal despite being implicitly mapped into a high dimensional space $\mathcal{H} = \mathbb{R}^H$. In particular, $H \gg N \times (D+1)$ makes KGCN clearly more efficient and still more effective compared to SGCN (see again table II and also table III and Fig. 3); this performance improves further as N (the number of learned support vectors $\{v_i^\theta\}_i$ per filter in Eq. 3) and K (the number of convolutional filters) reach reasonably (but not very) large values, and this results from the flexibility of the filters which learn — with few support vectors — relevant *representatives* of nodes in training data. These performances consistently improve for all the kernels and this is again explained by the representational power of the maps of these kernels. Moreover, the ablation study in table IV shows that KGCN with learned support vectors capture better the nodes in graph data while KGCN is clearly limited when the support vectors are fixed (and thereby biased i.e., not sufficiently representative of the actual distribution of the nodes, see again table IV); hence, learning the KGCN parameters (i.e., with learned α and fixed support vectors) is not enough in order to recover from this bias. In sum, the gain

GCNs / kernels	Standard GCN with different # of KPCA dimensions (H)										Our KGCN
	10	50	100	200	300	400	500	1000	2000	3000	
Linear	92.3077	overdim	overdim	overdim	overdim	overdim	overdim	overdim	overdim	overdim	90.7692
Poly	89.2308	95.3846	92.3077	93.8462	93.8462	93.8462	93.8462	overdim	overdim	overdim	93.8462
tanh	89.2308	93.8462	90.7692	93.8462	90.7692	92.3077	93.8462	92.3077	93.8462	92.3077	96.9231
sigmoid	93.8462	90.7692	93.8462	92.3077	92.3077	92.3077	92.3077	96.9231	93.8462	92.3077	95.3846
Gaussian	92.3077	92.3077	92.3077	92.3077	96.9231	93.8462	93.8462	93.8462	93.8462	93.8462	98.4615
Laplacian	92.3077	93.8462	95.3846	92.3077	90.7692	90.7692	95.3846	93.8462	90.7692	90.7692	98.4615
Power	90.7692	92.3077	95.3846	92.3077	92.3077	95.3846	95.3846	93.8462	93.8462	92.3077	96.9231
IMQ	87.6923	92.3077	95.3846	95.3846	93.8462	93.8462	90.7692	95.3846	93.8462	93.8462	95.3846
Log	93.8462	92.3077	92.3077	95.3846	93.8462	93.8462	95.3846	90.7692	95.3846	90.7692	96.9231
Cauchy	93.8462	95.3846	95.3846	92.3077	96.9231	93.8462	92.3077	95.3846	92.3077	93.8462	98.4615
HI	93.8462	92.3077	89.2308	90.7692	92.3077	92.3077	87.6923	87.6923	90.7692	87.6923	96.9231
time/epoch (s)	0.032	0.057	0.072	0.113	0.150	0.190	0.229	0.440	0.840	1.252	0.210

TABLE II

THIS TABLE SHOWS A COMPARISON OF OUR KGCN AGAINST SGCN (WITH DIFFERENT NUMBERS OF KPCA DIMENSIONS). NOTE THAT SGCN PERFORMANCES ARE NOT NECESSARILY INCREASING W.R.T H; INDEED, WHILE MORE DIMENSIONS CAPTURE MORE STATISTICAL VARIANCE, THIS ALSO INCREASES THE NUMBER OF TRAINING PARAMETERS AND HENCE THE RISK OF OVERFITTING. NOTE THAT FOR LINEAR AND POLYNOMIAL KERNELS, THE NUMBER OF DIMENSIONS (H) CANNOT EXCEED D AND D^2 RESPECTIVELY WITH $D = 24$ IN PRACTICE (THE KRONECKER TENSOR PRODUCT DEFINING THE MAP OF –ORDER 2– POLYNOMIAL KERNEL HAS D^2 DIMENSIONS WHILE THE MAP OF THE LINEAR KERNEL HAS OBVIOUSLY D DIMENSIONS.)

of our KGCN results from the *complementary aspects of the used (implicit) kernel maps and also the modeling capacity of our KGCN when the support vectors of these kernels (that define the convolutional filters) are also allowed to vary*.

# of Filters (K) \ # of SVs (N)	1	4	8
1	84.6154	84.7552	85.1748
5	93.1469	**95.3846**	92.8671
10	92.1678	95.1049	95.1049

TABLE III

AVERAGE ACCURACY (W.R.T ALL THE USED KERNELS IN KGCN) FOR DIFFERENT NUMBERS (K) AND SIZES (N) OF FILTERS. SVS STANDS FOR SUPPORT VECTORS.

Finally, we compare the classification performances of our KGCN against other related methods in action recognition ranging from sequence based such as LSTM and GRU [54]–[56] to deep graph (non-vectorial) methods based on spatial and spectral convolution [57]–[59]. From the results in table V, our KGCN brings a substantial gain w.r.t state of the art methods, and provides comparable results with the best vectorial methods.

IV. CONCLUSION

We introduce in this paper a novel GCN formulation based on kernel machines. The method defines convolutional graph filters in the span of nodes in a (high or potentially infinite dimensional) reproducing kernel Hilbert space (*RKHS*), with the particularity that node representations, in the *RKHS*, are learned instead of being taken from training data. This makes the proposed approach (semi-)parametric and tractable while also being effective and less subject to overfitting. Indeed, the proposed GCN formulation is dual and requires few parameters, it also provides an effective way to enhance the discrimination power of the learned graph representations and it overtakes standard (primal) GCN approaches as well as the related work.

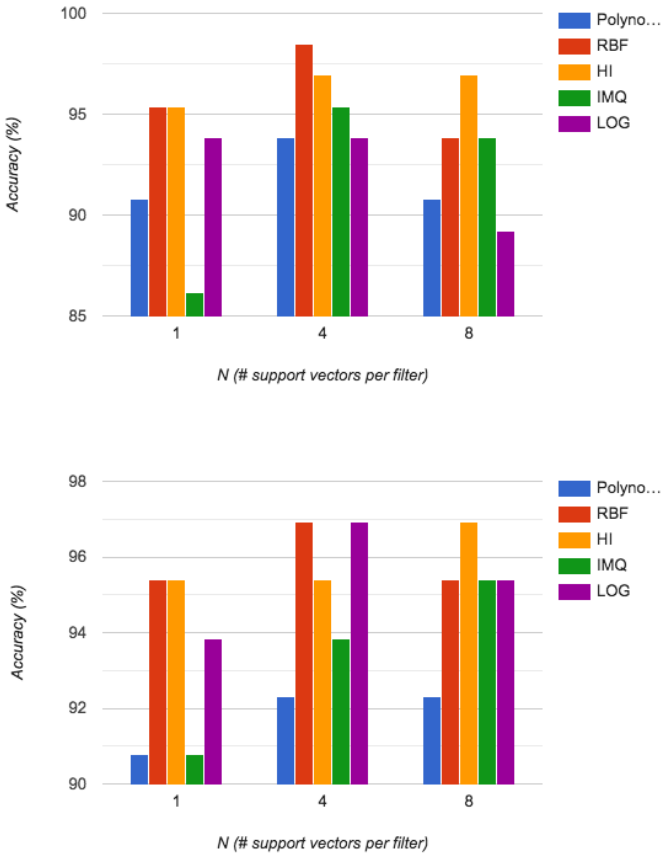

Fig. 3. Accuracy of KGCN w.r.t five examples of kernels and filter sizes N. Top figure corresponds to $K = 5$ filters while bottom one to $K = 10$. **(Best viewed in color)**.

KGCNs / kernels	F-SV / L-α	L-SV / F-α	L-SV / L-α
Linear	89.2308	90.7692	90.7692
Polynomial	84.6154	90.7692	93.8462
tanh	87.6923	90.7692	96.9231
Sigmoid	95.3846	95.3846	95.3846
Gaussian	84.6154	93.8462	98.4615
Laplacian	84.6154	93.8462	98.4615
Power	92.3077	95.3846	96.9231
I. Multi-quadric	81.5385	93.8462	95.3846
Log	84.6154	90.7692	96.9231
Cauchy	86.1538	92.3077	98.4615
HI	86.1538	95.3846	96.9231

TABLE IV

THIS TABLE SHOWS AN ABLATION STUDY; F-SV, F-α STAND RESPECTIVELY FOR FIXED SUPPORT VECTORS AND FIXED MIXING PARAMETERS α WHILE L-SV, L-α STAND FOR LEARNED ONES. NOTE THAT THE RESULTS, WHEN LEARNING THE SUPPORT VECTORS USING THE LINEAR KERNEL, ARE IDENTICAL (BOTH WITH FIXED AND LEARNED α) AS ONE MAY INCLUDE THE MULTIPLICATIVE FACTORS α IN THE LEARNED SUPPORT VECTORS (THE CONVERSE IS NOT TRUE).

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

TABLE V

COMPARISON AGAINST STATE OF THE ART METHODS.

As a future work, we are currently investigating the combination of explicit node expansion with implicit kernel mapping, in order to further enhance the generalization performances of other pattern recognition tasks.

REFERENCES

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

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

[3] Atwood, J., Towsley, D.: Diffusion-convolutional neural networks. In: Advances in Neural Information Processing Systems. pp. 1993–2001 (2016)

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

[5] Burges, C.J., Schölkopf, B.: Improving the accuracy and speed of support vector machines. In: Advances in neural information processing systems. pp. 375–381 (1997)

[6] Chen, J., Zhu, J., Song, L.: Stochastic training of graph convolutional networks with variance reduction. arXiv preprint arXiv:1710.10568 (2017)

[7] N. Boujemaa, F. Fleuret, V. Gouet, and H. Sahbi (2004, January). Visual content extraction for automatic semantic annotation of video news. In the proceedings of the SPIE Conference, San Jose, CA (Vol. 6).

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

[9] Cho, Y., Saul, L.K.: Kernel methods for deep learning. In: Advances in neural information processing systems. pp. 342–350 (2009)

[10] Cortes, C., Mohri, M., Rostamizadeh, A.: Learning non-linear combinations of kernels. In: Advances in neural information processing systems. pp. 396–404 (2009)

[11] Dai, B., Xie, B., He, N., Liang, Y., Raj, A., Balcan, M.F.F., Song, L.: Scalable kernel methods via doubly stochastic gradients. In: Advances in Neural Information Processing Systems. pp. 3041–3049 (2014)

[12] Dai, H., Kozareva, Z., Dai, B., Smola, A., Song, L.: Learning steady-states of iterative algorithms over graphs. In: International Conference on Machine Learning. pp. 1114–1122 (2018)

[13] H. Sahbi and N. Boujemaa. "From coarse to fine skin and face detection." Proceedings of the eighth ACM international conference on Multimedia. 2000.

[14] Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: Advances in neural information processing systems. pp. 3844–3852 (2016)

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

[16] Genton, M.G.: Classes of kernels for machine learning: a statistics perspective. Journal of machine learning research **2**(Dec), 299–312 (2001)

[17] Gönen, M., Alpaydın, E.: Multiple kernel learning algorithms. Journal of machine learning research **12**(Jul), 2211–2268 (2011)

[18] Gori, M., Monfardini, G., Scarselli, F.: A new model for learning in graph domains. In: Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005. vol. 2, pp. 729–734. IEEE (2005)

[19] Hamid, R., Xiao, Y., Gittens, A., DeCoste, D.: Compact random feature maps. In: International Conference on Machine Learning. pp. 19–27 (2014)

[20] Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems. pp. 1024–1034 (2017)

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

[22] Huang, W., Zhang, T., Rong, Y., Huang, J.: Adaptive sampling towards fast graph representation learning. In: Advances in Neural Information Processing Systems. pp. 4558–4567 (2018)

[23] Kar, P., Karnick, H.: Random feature maps for dot product kernels. In: Artificial Intelligence and Statistics. pp. 583–591 (2012)

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

[25] H. Sahbi and F. Fleuret. "Scale-invariance of support vector machines based on the triangular kernel." (2002).

[26] Le, Q., Sarlós, T., Smola, A.: Fastfood-approximating kernel expansions in loglinear time. In: Proceedings of the international conference on machine learning. vol. 85 (2013)

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

[28] Li, R., Wang, S., Zhu, F., Huang, J.: Adaptive graph convolutional neural networks. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)

[29] Li, Y., Tarlow, D., Brockschmidt, M., Zemel, R.: Gated graph sequence neural networks. arXiv preprint arXiv:1511.05493 (2015)

[30] Lu, Z., May, A., Liu, K., Garakani, A.B., Guo, D., Bellet, A., Fan, L., Collins, M., Kingsbury, B., Picheny, M., et al.: How to scale up kernel methods to be as good as deep neural nets. arXiv preprint arXiv:1411.4000 (2014)

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

[32] Maji, S., Berg, A.C., Malik, J.: Efficient classification for additive kernel svms. IEEE transactions on pattern analysis and machine intelligence **35**(1), 66–77 (2012)

[33] Micheli, A.: Neural network for graphs: A contextual constructive approach. IEEE Transactions on Neural Networks **20**(3), 498–511 (2009)

[34] Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., Bronstein, M.M.: Geometric deep learning on graphs and manifolds using mixture model cnns. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5115–5124 (2017)

[35] Niepert, M., Ahmed, M., Kutzkov, K.: Learning convolutional neural networks for graphs. In: International conference on machine learning. pp. 2014–2023 (2016)

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

[37] Rahimi, A., Recht, B.: Random features for large-scale kernel machines. In: Advances in neural information processing systems. pp. 1177–1184 (2008)

[38] Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. IEEE Transactions on Neural Networks **20**(1), 61–80 (2008)
Maji, S., Berg, A.C., Malik, J.: Efficient classification for additive kernel svms. IEEE transactions on pattern analysis and machine intelligence **35**(1), 66–77 (2012)

[39] Vedaldi, A., Zisserman, A.: Efficient additive kernels via explicit feature maps. IEEE transactions on pattern analysis and machine intelligence **34**(3), 480–492 (2012)

[40] Vo, P., Sahbi, H.: Transductive kernel map learning and its application to image annotation. In: BMVC. pp. 1–12 (2012)

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

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

[43] Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. Deep learning. MIT press, 2016.

[44] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

[45] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[46] H. Sahbi, Coarse to fine support vector machines for hierarchical face detection, Ph.D. thesis, PhD thesis, Versailles University, 2003.

[47] Howard, Andrew G., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." arXiv preprint arXiv:1704.04861 (2017).

[48] Iandola, Forrest N., et al. "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and¡ 0.5 MB model size." arXiv preprint arXiv:1602.07360 (2016).

[49] Szegedy, Christian, et al. "Going deeper with convolutions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[50] Shawe-Taylor, John, and Nello Cristianini. Kernel methods for pattern analysis. Cambridge university press, 2004.

[51] H. Sahbi and F. Fleuret. "Kernel methods and scale invariance using the triangular kernel." (2004).

[52] Shahroudy, Amir, et al. "Ntu rgb+ d: A large scale dataset for 3d human activity analysis." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[53] Kiwon Yun, Jean Honorio, Debaleena Chattopadhyay, Tamara L. Berg, and Dimitris Samaras, The 2nd International Workshop on Human Activity Understanding from 3D Data at Conference on Computer Vision and Pattern Recognition (HAU3D-CVPRW), CVPR 2012

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

[55] 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

[56] 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

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

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

[59] 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

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

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

[62] 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

[63] 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

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

[65] H. Sahbi. "A particular Gaussian mixture model for clustering and its application to image retrieval." Soft Computing 12.7 (2008): 667-676.

[66] 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

[67] 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

[68] 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

[69] 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

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

[71] Zhuang, Chenyi, and Qiang Ma. "Dual graph convolutional networks for graph-based semi-supervised classification." Proceedings of the 2018 World Wide Web Conference. 2018.

[72] Bacciu, Davide, Federico Errica, and Alessio Micheli. "Contextual graph markov model: A deep and generative approach to graph processing." arXiv preprint arXiv:1805.10636 (2018).

[73] Q. Oliveau 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.

[74] Zhang, Muhan, et al. "An end-to-end deep learning architecture for graph classification." Thirty-Second AAAI Conference on Artificial Intelligence. 2018.

[75] Ying, Zhitao, et al. "Hierarchical graph representation learning with differentiable pooling." Advances in neural information processing systems. 2018.

[76] Liu, Ziqi, et al. "Geniepath: Graph neural networks with adaptive receptive paths." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 33. 2019.

[77] Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2018). How powerful are graph neural networks?. arXiv preprint arXiv:1810.00826.

[78] Kipf, Thomas N., and Max Welling. "Variational graph auto-encoders." arXiv preprint arXiv:1611.07308 (2016).

[79] Schölkopf, Bernhard, Ralf Herbrich, and Alex J. Smola. "A generalized representer theorem." International conference on computational learning theory. Springer, Berlin, Heidelberg, 2001.

[80] G.S. Kimeldorf and G. Wahba. A correspondence between Bayesian estimation on stochastic processes and smoothing by splines. Annals of Mathematical Statistics, 41:495–502, 1970.

[81] Vapnik, Vladimir N. "An overview of statistical learning theory." IEEE transactions on neural networks 10.5 (1999): 988-999.