TOPOLOGICALLY-CONSISTENT MAGNITUDE PRUNING FOR VERY LIGHTWEIGHT GRAPH CONVOLUTIONAL NETWORKS

Hichem Sahbi

TOPOLOGICALLY-CONSISTENT MAGNITUDE PRUNING FOR VERY LIGHTWEIGHT GRAPH CONVOLUTIONAL NETWORKS

Hichem Sahbi

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

ABSTRACT

Graph convolution networks (GCNs) are currently mainstream in learning with irregular data. These models rely on message passing and attention mechanisms that capture context and node-to-node relationships. With multi-head attention, GCNs become highly accurate but oversized, and their deployment on cheap devices requires their pruning. However, pruning at high regimes usually leads to topologically inconsistent networks with weak generalization.
In this paper, we devise a novel method for lightweight GCN design. Our proposed approach parses and selects subnetworks with the highest magnitudes while guaranteeing their topological consistency. The latter is obtained by selecting only accessible and co-accessible connections which actually contribute in the evaluation of the selected subnetworks. Experiments conducted on the challenging FPHA dataset show the substantial gain of our topologically consistent pruning method especially at very high pruning regimes.

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

1. INTRODUCTION

Deep convolutional networks are currently one of the most successful models in image processing and pattern recognition [19]. Their principle consists in learning convolutional filters, together with attention and fully connected layers, that maximize classification performances. These models are mainly suitable for data sitting on top of regular domains (such as images) [2, 8, 19, 40], but their adaptation to irregular data (namely graphs) requires extending convolutions to arbitrary domains [27, 28, 49]; these extensions are known as graph convolutional networks (GCNs).

Two categories of GCNs exist in the literature: spectral and spatial. Spectral methods [29–33] proceed by projecting both input graph signals and convolutional filters using the Fourier transform, and achieve convolution in the Fourier domain, prior to back-project the resulting convolved signal in the input domain. These projections rely on the eigen-decomposition of graph Laplacians whose complexity scales polynomially with the size of the input graphs [15,23,38], and this makes spectral GCNs clearly intractable. Spatial methods [34–37] instead rely on message passing, via attention matrices, before applying convolution. While spatial GCNs have been relatively more effective compared to spectral ones, their success is highly reliant on the accuracy of the attention matrices that capture context and node-to-node relationships [39]. With multi-head attention, GCNs are more accurate but computationally more demanding, so lightweight variants of these models should instead be considered.

Several methods have been proposed in the literature in order to design lightweight yet effective deep convolutional networks [42–45]. Some of them build efficient networks from scratch while others pretrain heavy networks prior to reduce their time and memory footprint using distillation [46–48, 50–52] and pruning [53–55]. Pruning methods, either unstructured or structured, allow removing connections whose impact on the classification performance is the least perceptible. Unstructured pruning [55, 56] consists in cutting connections individually using different criteria, including weight magnitude[1], prior to fine-tuning. In contrast, structured pruning [57, 59] aims at removing groups of connections, channels or entire subnetworks. Whereas structured pruning may reach high speed-up on dedicated hardware resources, its downside resides in the rigidity of the class of learnable lightweight networks. On another side, unstructured pruning is more flexible, but may result into *topologically inconsistent* subnetworks (i.e., either partially or completely disconnected), and this may lead to limited generalization especially at very high pruning rates.

In this paper, we introduce a novel approach for lightweight GCN design that gathers the advantage of both structured and unstructured pruning, and discards their inconvenient; i.e., the method imposes a few constraints on the structure of the learned subnetworks (namely their topological consistency) while also ensuring their flexibility at some extent. Our solution is greedy and proceeds by selecting connections with the highest magnitudes while guaranteeing their *accessibility* (i.e., their reachability from the network input) and their *co-accessibility* (i.e., their actual contribution in the evaluation of the output). Hence, only topologically consistent subnetworks are considered when selecting connections. Different magnitude surrogates are also considered in order to greedily

[1] Magnitude is considered as a proxy of weight relevance.

select connections; these surrogates allow maximizing magnitudes not only locally at the visited layers but also globally at the subsequent ones, thereby resulting into more effective lightweight networks as shown in experiments.

2. 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 a signal $\{\psi(u) \in \mathbb{R}^s : u \in \mathcal{V}\}$ and associated with an adjacency matrix $\mathbf{A}$ with each entry $\mathbf{A}_{uu'} > 0$ iff $(u, u') \in \mathcal{E}$ and 0 otherwise. GCNs aim at learning a set of C filters $\mathcal{F}$ that define convolution on n nodes of $\mathcal{G}$ (with $n = |\mathcal{V}|$) as

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

here $^\top$ stands for transpose, $\mathbf{U} \in \mathbb{R}^{s \times n}$ is the graph signal, $\mathbf{W} \in \mathbb{R}^{s \times C}$ is the matrix of convolutional parameters corresponding to the C filters and $f(.)$ is a nonlinear activation applied entrywise. In Eq. 1, the input signal $\mathbf{U}$ is projected using $\mathbf{A}$ and this provides for each node u, the aggregate set of its neighbors. Entries of $\mathbf{A}$ could be handcrafted or learned so Eq. 1 implements a convolutional block with two layers; the first one aggregates signals in $\mathcal{N}(\mathcal{V})$ (sets of node neighbors) by multiplying $\mathbf{U}$ with $\mathbf{A}$ while the second layer achieves convolution by multiplying the resulting aggregates with the C filters in $\mathbf{W}$. Learning multiple adjacency (also referred to as attention) matrices (denoted as $\{\mathbf{A}^k\}_{k=1}^K$) allows capturing different contexts and graph topologies when achieving aggregation and convolution. With multiple matrices $\{\mathbf{A}^k\}_k$ (and associated convolutional filter parameters $\{\mathbf{W}^k\}_k$), Eq. 1 is updated as $(\mathcal{G} \star \mathcal{F})_{\mathcal{V}} = f\big(\sum_{k=1}^K \mathbf{A}^k \mathbf{U}^\top \mathbf{W}^k\big)$. Stacking aggregation and convolutional layers, with multiple matrices $\{\mathbf{A}^k\}_k$, makes GCNs accurate but heavy. In what follows, we propose a novel method that makes our networks lightweight and still effective.

3. LIGHTWEIGHT DESIGN

In what follows, we subsume any given GCN as a multi-layered neural network g_θ whose weights defined as $\theta = \big\{\mathbf{W}^1, \ldots, \mathbf{W}^L\big\}$, with L being its depth, $\mathbf{W}^\ell \in \mathbb{R}^{d_{\ell-1} \times d_\ell}$ its ℓ^{th} layer weights, and d_ℓ the dimension of ℓ. The output of a given layer ℓ is defined as

$$\phi^\ell = f_\ell(\mathbf{W}^{\ell\,\top}\ \phi^{\ell-1}), \quad \ell \in \{1, \ldots, L-1\}, \tag{2}$$

being f_ℓ an activation function. Without a loss of generality, we omit the bias in the definition of (2).

3.1. Magnitude Pruning

Given a GCN g_θ, magnitude pruning (MP) consists in removing connections in g_θ. MP is obtained by zeroing-out a subset of weights in θ, and this is achieved by multiplying $\mathbf{W}^\ell$ by a binary mask $\mathbf{M}^\ell \in \{0,1\}^{d_{\ell-1} \times d_\ell}$. The binary entries of $\mathbf{M}^\ell$ are set depending on whether the underlying layer connections are kept or removed, so Eq. 2 becomes

$$\phi^\ell = f_\ell((\mathbf{M}^\ell \odot \mathbf{W}^\ell)^\top\ \phi^{\ell-1}), \tag{3}$$

here $\odot$ stands for the element-wise matrix product. In this definition, entries of the tensor $\{\mathbf{M}^\ell\}_\ell$ are set depending on the prominence of the underlying connections in g_θ; MP consists first in zeroing the smallest parameters (up to a pruning rate) in the learned GCN g_θ, and then fine-tuning the remaining parameters. However, such MP suffers from several drawbacks. On the one hand, removing connections individually may result into *topologically inconsistent* networks (see section 3.2), i.e., either completely disconnected or having isolated connections. On the other hand, high pruning rates may lead to an over-regularization effect and hence weakly discriminant lightweight networks, especially when the latter include isolated connections (see later experiments). In what follows, we introduce a more principled MP that guarantees the topological consistency of the pruned networks and allows improving generalization even at very high pruning rates.

3.2. Our Topologically Consistent Magnitude Pruning

Our formal definition of topological consistency relies on two principles: *accessibility and co-accessibility* of connections in g_θ. Let $\mathbf{M}^\ell_{i,j}$ refer to a connection between the i-th and the j-th neurons of layer ℓ. $\mathbf{M}^\ell_{i,j}$ is accessible if $\exists i_1, \ldots, i_{\ell-1}$, s.t. $\mathbf{M}^1_{i_1,i_2} = \cdots = \mathbf{M}^{\ell-1}_{i_{\ell-1},i} = 1$, and $\mathbf{M}^\ell_{i,j}$ is co-accessible if $\exists i_{\ell+1}, \ldots, i_L$, s.t. $\mathbf{M}^{\ell+1}_{j,i_{\ell+1}} = \cdots = \mathbf{M}^L_{i_{L-1},i_L} = 1$. Considering the products $\mathbf{S}^\ell_a = \mathbf{M}^1\ \mathbf{M}^2 \ldots \mathbf{M}^{\ell-1}$ and $\mathbf{S}^\ell_c = \mathbf{M}^{\ell+1}\ \mathbf{M}^{\ell+2} \ldots \mathbf{M}^L$, and following the above definition, it is easy to see that $\mathbf{M}^\ell_{ij}$ is accessible (resp. co-accessible) iff the i-th column (resp. j-th row) of $\mathbf{S}^\ell_a$ (resp. $\mathbf{S}^\ell_c$) is different from the null vector. A network is called topologically consistent iff all its connections are both accessible and co-accessible. Accessibility guarantees that incoming connections to the i-th neuron carry out effective activations resulting from the evaluation of g_θ up to layer ℓ. Co-accessibility is equivalently important and guarantees that outgoing activation from the j-th neuron actually contributes in the evaluation of the network output. A connection $\mathbf{M}^\ell_{ij}$ — not satisfying accessibility or co-accessibility and even when its magnitude is large — becomes useless and should be removed when g_θ is pruned.

For any given network, parsing all its topologically consistent subnetworks and keeping only the one with the largest magnitudes is highly combinatorial. Indeed, the accessibility of a given connection depends on whether its preceding and subsequent ones are kept or removed, and any masked connections may affect the accessibility of the others. In what follows, we introduce a greedy algorithm that prunes a given network by maximizing the magnitude of its connections while guaranteeing its topological consistency.

Algorithm 1: Topologically consistent MP

Input: Weight tensor $\{\mathbf{W}^1, \dots, \mathbf{W}^L\}$, MaxKeptConnections.
Output: Mask tensor $\{\mathbf{M}^1, \dots, \mathbf{M}^L\}$.

$nc \leftarrow 0$; $\{\mathbf{M}^\ell \leftarrow 0\}_\ell$;
while $nc <$ *MaxKeptConnections* **do**
  Select i_1 from $\{1, \dots, d_1\}$;
  for $\ell = 1$ **to** $L-1$ **do**
    $i_{\ell+1} \leftarrow \arg\max_{j \in \mathcal{N}_\ell(i_\ell),k} [\mathbf{W}^\ell_{i_\ell,j} \hat{\mathbf{W}}^{\ell+1}_{j,k}]$; // A stochastic variant is to select a random walk from i_ℓ to $i_{\ell+1} \in \mathcal{N}_\ell(i_\ell)$ proportionally to $\{\mathbf{W}^\ell_{i_\ell,j} \hat{\mathbf{W}}^{\ell+1}_{j,k}\}_{j \in \mathcal{N}_\ell(i_\ell),k}$.
    if $(\mathbf{M}^\ell_{i_\ell,i_{\ell+1}} = 0)$ **then**
      $nc \leftarrow nc + 1$;
    $\mathbf{M}^\ell_{i_\ell,i_{\ell+1}} \leftarrow 1$;

3.3. Algorithm

Our solution parses neurons in g_θ layer-wise; each parsing consists in finding a complete chain from the input to the output of g_θ. Given a neuron i in layer ℓ, its subsequent neuron in the chain corresponds to the one which maximizes magnitude among the forward neighbors of i (denoted as $\mathcal{N}_\ell(i)$). This process is repeated for different input neurons till exhausting a targeted pruning rate. This solution maximizes magnitude locally; however, there is not guarantee that neurons visited in the subsequent layers will have sufficiently large magnitude connections. In order to circumvent this issue, instead of locally maximizing $\{\mathbf{W}^\ell_{i,j}\}_{j \in \mathcal{N}_\ell(i)}$, we *globally* maximize a surrogate criterion as

$$\max_{j \in \mathcal{N}_\ell(i),k} \mathbf{W}^\ell_{i,j} \hat{\mathbf{W}}^{\ell+1}_{j,k}, \quad k \in \{1, \dots, d_L\}, \tag{4}$$

with $\hat{\mathbf{W}}^{\ell+1} = \mathbf{W}^{\ell+1} \dots \mathbf{W}^L$ (see also algorithm 1). Under row-stochasticity of $\{\mathbf{W}^\ell\}_\ell$, the matrix $\hat{\mathbf{W}}^{\ell+1}$ models an m-step markovian process (with $m = L - \ell$) where the conditional transition likelihood, between two neurons, is proportional to the sum of the conditional likelihoods of all the possible $m-1$ steps linking these two neurons. Nevertheless, Eq. 4 could be contaminated by a large number of small magnitude connections. This limitation motivates the introduction of a slight variant (called α-powered magnitude) with $\hat{\mathbf{W}}^{\ell+1}$ recursively defined as

$$\hat{\mathbf{W}}^{\ell+1} = \left[[\mathbf{W}^{\ell+1}]^{\frac{1}{\alpha}} [\hat{\mathbf{W}}^{\ell+2}]^{\frac{1}{\alpha}} \right]^\alpha, \quad 1/\alpha \in [1, +\infty[, \tag{5}$$

here the power is applied entrywise. When $\alpha \to 0$, Eq. 4 captures the *largest magnitude path*[2] outgoing from the i-th to the k-th output neuron of g_θ (via j). When $\alpha \in]0, 1[$, Eq. 4 models instead *the average of dominating magnitude paths* outgoing from the i-th neuron (again via j), so the effect of spurious (small magnitude) connections could be attenuated.

[2]The magnitude of a path is defined as the sum of all its connection magnitudes.

3.4. Stochasticity

In spite of making the selected subnetworks topologically consistent and hence effective (as shown later in experiments), the aforementioned procedure is deterministic and relies on the hypothesis that only connections with the highest magnitudes are essential while in practice, other connections could also be used in order to *explore* further subnetworks. Hence, instead of considering a deterministic parsing approach, we consider a stochastic sampling process. More precisely, neurons are again visited layer-wise, but the subsequent layer neurons are selected by sampling a random walk distribution (see the commented variant in algorithm 1); note that neurons that maximize magnitude are still preferred (with a high probability), nevertheless other neurons will also be selected depending on their magnitude distribution. This stochastic variant turns out to be more effective, especially at high pruning regimes, as shown subsequently.

4. EXPERIMENTS

We evaluate the performance of our GCNs on the task of action recognition using the First-Person Hand Action (FPHA) dataset [3]. The latter includes 1175 skeletons belonging to 45 action categories (with style, speed, scale and viewpoint variations). Each video (sequence of skeletons) is initially described with a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where each node $v_j \in \mathcal{V}$ corresponds to the j-th hand-joint trajectory (denoted as $\{\hat{p}^t_j\}_t$) and an edge $(v_j, v_i) \in \mathcal{E}$ exists iff the j-th and the i-th trajectories are spatially connected. Each trajectory in $\mathcal{G}$ is processed using *temporal chunking* [26]: first, the total duration of a sequence is split into M equally-sized temporal chunks ($M = 32$ in practice), then the trajectory coordinates $\{\hat{p}^t_j\}_t$ are assigned to the M chunks (depending on their time stamps) prior to concatenate the averages of these chunks. This produces the raw description (signal) of v_j.

Implementation details and baseline GCN. We trained the GCNs end-to-end using the Adam optimizer [1] for 2,700 epochs with a batch size equal to 600, a momentum of 0.9 and a global learning rate (denoted as $\nu(t)$) inversely proportional to the speed of change of the cross entropy loss used to train our networks. When this speed increases (resp. decreases), $\nu(t)$ decreases as $\nu(t) \leftarrow \nu(t-1) \times 0.99$ (resp. increases as $\nu(t) \leftarrow \nu(t-1)/0.99$). In all these experiments, we use a GeForce GTX 1070 GPU (with 8 GB memory). We evaluate the performances using the 1:1 setting proposed in [3] with 600 action sequences for training and 575 for testing, and we report the average accuracy over all the classes of actions. The architecture of our baseline GCN (taken from [58]) includes an attention layer of 16 heads applied to skeleton graphs whose nodes are encoded with 32-channels, followed by a convolutional layer of 128 filters, and a dense fully connected layer. In total, this initial network is relatively heavy

(for a GCN) and its number of parameters reaches 2 millions. Nevertheless, this GCN is accurate compared to the related work on the FPHA benchmark as shown in Table. 1. Considering this GCN baseline, our goal is to make it lightweight while maintaining its high accuracy.

Method	Color	Depth	Pose	Accuracy (%)
Two stream-color [4]	✓	✗	✗	61.56
Two stream-flow [4]	✓	✗	✗	69.91
Two stream-all [4]	✓	✗	✗	75.30
HOG2-depth [5]	✗	✓	✗	59.83
HOG2-depth+pose [5]	✗	✓	✓	66.78
HON4D [6]	✗	✓	✗	70.61
Novel View [7]	✗	✓	✗	69.21
1-layer LSTM [9]	✗	✗	✓	78.73
2-layer LSTM [9]	✗	✗	✓	80.14
Moving Pose [10]	✗	✗	✓	56.34
Lie Group [11]	✗	✗	✓	82.69
HBRNN [12]	✗	✗	✓	77.40
Gram Matrix [13]	✗	✗	✓	85.39
TF [14]	✗	✗	✓	80.69
JOULE-color [16]	✓	✗	✗	66.78
JOULE-depth [16]	✗	✓	✗	60.17
JOULE-pose [16]	✗	✗	✓	74.60
JOULE-all [16]	✓	✓	✓	78.78
Huang et al. [17]	✗	✗	✓	84.35
Huang et al. [18]	✗	✗	✓	77.57
Our GCN baseline	✗	✗	✓	**86.08**

Table 1: Comparison of our baseline GCN against related work on FPHA.

Lightweight CGN performances. Table. 2 shows the accuracy of our lightweight GCNs for different pruning rates, and other settings including topological consistency (TC) and stochasticity. From these results, the impact of TC is substantial on highly pruned GCNs. We also observe the positive impact of stochasticity which allows exploring different subnetworks. Note that the impact of TC is less important (and sometimes worse) with low pruning regimes; indeed, low pruning rates produce subnetworks with already enough (a large number of) connections and *having some of them neither accessible nor co-accessible* produces a well known regularization effect [41]. On another side, over-pruning networks, without TC, produces an over-regularization effect (i.e., under-fitting); the resulting lightweight networks become highly disconnected. In contrast, TC ensures connectivity (accessibility and co-accessibility) in spite of learning very lightweight networks, it also mitigates under-fitting and thereby improves generalization (see again accuracy in Table. 2 and how the A-C difference between standard MP and TC MP is accentuated as pruning rates increase). Finally, Table. 3 shows the impact of α (in Eq. 5) on the performance of our lightweight GCNs. From this table, sufficiently (but not very) large α makes accuracy improving; as m-step magnitude estimation takes into account the dominating magnitude paths, it is more robust.

Pruning rates	TC	Stochastic	# parameters	% of A-C	Accuracy (%)	Observation
0 %	NA	NA	1967616	100	86.08	Baseline GCN
50%	✗	✗	983808	100	86.08	Standard MP
	✗	✓		100	86.08	Stochastic MP
	✓	✗		100	86.08	TC MP
	✓	✓		100	86.08	TC Stoch MP
75%	✗	✗	491904	99.4	85.73	Standard MP
	✗	✓		99.8	**85.91**	Stochastic MP
	✓	✗		100	84.86	TC MP
	✓	✓		100	**85.91**	TC Stoch MP
90%	✗	✗	196760	89.9	85.04	Standard MP
	✗	✓		92.3	**85.56**	Stochastic MP
	✓	✗		100	83.65	TC MP
	✓	✓		100	**85.56**	TC Stoch MP
95%	✗	✗	98379	72.3	83.82	Standard MP
	✗	✓		76.5	85.39	Stochastic MP
	✓	✗		100	**85.73**	TC MP
	✓	✓		100	84.86	TC Stoch MP
99%	✗	✗	19674	21.2	76.00	Standard MP
	✗	✓		28.2	74.08	Stochastic MP
	✓	✗		100	**83.47**	TC MP
	✓	✓		100	80.69	TC Stoch MP
99.9%	✗	✗	1966	1.2	2.78	Standard MP
	✗	✓		0.0	NA	Disconnected
	✓	✗		100	70.08	TC MP
	✓	✓		100	**73.39**	TC Stoch MP

Table 2: Detailed performances and ablation study, for different pruning rates (# of parameters) and other criteria including topological consistency (TC) and stochasticity. This table also shows the resulting percentage of accessible and co-accessible connections (denoted as A-C). These results are obtained by maximizing magnitudes locally (without Eqs. 4 and 5, i.e., by maximizing $\{\mathbf{W}_{i,j}^{\ell}\}_{j\in\mathcal{N}_{\ell}(i)}$ instead). NA stands for not applicable.

$1/\alpha$	1	1.5	2.5	7	10	20	50
Accuracy (%)	69.56	71.82	72.34	73.21	**76.17**	72.86	66.60

Table 3: Accuracy for different α settings when maximizing magnitudes globally (i.e., with Eqs. 4 and 5). Both TC and stochasticity are used, and pruning rate (PR) is set to 99.9 %. Compared to table 2, with the same PR, the accuracy improves significantly when α is set appropriately ($\alpha = 10$).

5. CONCLUSION

In this paper, we introduce a novel pruning method that trains very lightweight GCNs while guaranteeing their topological consistency. The latter is an important property which guarantees the contribution of all accessible and co-accessible network connections in the learned decision functions. Experiments conducted on the challenging task of hand-gesture recognition shows the maintained high accuracy of our topologically consistent lightweight GCNs, even at *very high* pruning regimes. As a future work, we are currently investigating the extension of this method to other network architectures and datasets.

6. REFERENCES

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

[2] M. Jiu and H. Sahbi. "Laplacian deep kernel learning for image annotation." IEEE ICASSP, 2016.

[3] G. Garcia-Hernando, S. Yuan, S. Baek, and T.-K. Kim. First- Person

Hand Action Benchmark with RGB-D Videos and 3D Hand Pose Annotations. In CVPR, 2018

[4] C. Feichtenhofer et al. Convolutional Two-Stream Network Fusion for Video Action Recognition. CVPR, pages 1933-1941, 2016. 8

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

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

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

[8] H. Sahbi et al.. "Context-dependent kernel design for object matching and recognition." In IEEE CVPR, pp. 1-8, 2008.

[9] W. Zhu et al. Co-occurrence feature learning for skeleton based action recognition using regularized deep LSTM networks In AAAI, volume 2, page 6, 2016.

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

[11] R. Vemulapalli et al. Human action recognition by representing 3D skeletons as points in a Lie group. In IEEE CVPR, pages 588–595, 2014

[12] Y. Du, W. Wang, and L. Wang. Hierarchical recurrent neural network for skeleton based action recognition. In IEEE CVPR, pages 1110–1118, 2015.

[13] X. Zhang et al. Efficient Temporal Sequence Comparison and Classification Using Gram Matrix Embeddings on a Riemannian Manifold. In CVPR, pages 4498–4507, 2016

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

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

[16] J. Hu et al. Jointly Learning Heterogeneous Features for RGB-D Activity Recognition. In CVPR, pages 5344-5352, 2015

[17] Z. Huang and L. V. Gool. A Riemannian Network for SPD Matrix Learning. In AAAI, pages 2036–2042, 2017

[18] Z. Huang, J. Wu, and L. V. Gool. Building Deep Networks on Grassmann Manifolds. In AAAI, pages 3279–3286, 2018

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

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[21] G. Huang, Z. Liu, L. v. d. Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *CVPR*, 2017, pp. 2261–2269.

[22] He, Kaiming, et al. "Mask r-cnn." Proceedings of ICCV, 2017.

[23] Q. Oliveau and H. Sahbi. "Learning attribute representations for remote sensing ship category classification." IEEE JSTARS, 10(6), 2017.

[24] Zhang, Ziwei, Peng Cui, and Wenwu Zhu. "Deep learning on graphs: A survey." IEEE TKDE (2020).

[25] O. Ronneberger et al. "U-net: Convolutional networks for biomedical image segmentation." International Conference on Medical image computing and computer-assisted intervention. Springer, 2015.

[26] A. Mazari and H. Sahbi. "MLGCN: Multi-Laplacian graph convolutional networks for human action recognition." BMVC, 2019.

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

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

[29] TN. Kipf, M. Welling. Semi-supervised classification with graph convolutional networks. In ICLR, 2017

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

[31] R. Li, S. Wang, F. Zhu, J. Huang. Adaptive graph convolutional neural networks. In AAAI, 2018.

[32] M. Defferrard et al. Convolutional Neural Networks on graphs with Fast Localized Spectral Filtering. In NIPS, 2016

[33] H. Sahbi. "Learning laplacians in chebyshev graph convolutional networks." Proceedings of the IEEE/CVF ICCV, 2021.

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

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

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

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

[38] Chung, Fan RK, and Fan Chung Graham. Spectral graph theory. No. 92. American Mathematical Soc., 1997.

[39] Knyazev et al. "Understanding attention and generalization in graph neural networks." In NIPS 2019.

[40] H. Sahbi. "Imageclef annotation with explicit context-aware kernel maps." IJMIR, 4.2 (2015): 113-128.

[41] Wan, Li, et al. "Regularization of neural networks using dropconnect." International conference on machine learning. PMLR, 2013.

[42] G. Huang et al. "Condensenet: An efficient densenet using learned group convolutions," in *CVPR*, 2018.

[43] M. Sandler et al. "Mobilenetv2: Inverted residuals and linear bottlenecks," in *CVPR*, 2018.

[44] A. G. Howard et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arxiv preprint*, 2017.

[45] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *ICML*. 2019, vol. 97, PMLR.

[46] G. E. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *CoRR*, vol. abs/1503.02531, 2015.

[47] S. Zagoruyko and N. Komodakis, "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer," in *ICLR*, 2017.

[48] A. Romero, N. Ballas, S. Ebrahimi Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," in *ICLR*, 2015.

[49] H. Sahbi. "Kernel-based Graph Convolutional Networks." 2020 25th International Conference on Pattern Recognition (ICPR). IEEE, 2021.

[50] S.-I. Mirzadeh et al. "Improved knowledge distillation via teacher assistant," in *AAAI*, 2020.

[51] Y. Zhang, T. Xiang, T. M. Hospedales, and H. Lu, "Deep mutual learning," in *CVPR*, 2018.

[52] S. Ahn, S. X. Hu, A. C. Damianou, N. D. Lawrence, and Z. Dai, "Variational information distillation for knowledge transfer," in *CVPR*, 2019.

[53] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *NIPS*, 1989.

[54] B. Hassibi and D. G. Stork, "Second order derivatives for network pruning: Optimal brain surgeon," in *NIPS*, 1992.

[55] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural network," in *NIPS*, 2015.

[56] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," in *ICLR*, 2016.

[57] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," in *ICLR*, 2017.

[58] H. Sahbi. "Learning Connectivity with Graph Convolutional Networks." International Conference on Pattern Recognition (ICPR), 2021.

[59] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *ICCV*. 2017, IEEE Computer Society.