SURVEY OF IMAGE BASED GRAPH NEURAL NETWORKS

U. Nazir[⋆†] *H. Wang*[⋆†] *M. Taj*[⋆]

[⋆†] {scun, H.E.Wang}@leeds.ac.uk
[⋆]{murtaza.taj}@lums.edu.pk

ABSTRACT

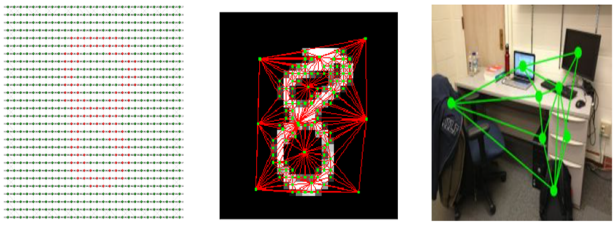
In this survey paper, we analyze image based graph neural networks and propose a three-step classification approach. We first convert the image into superpixels using the Quickshift algorithm so as to reduce 30% of the input data. The superpixels are subsequently used to generate a region adjacency graph. Finally, the graph is passed through a state-of-art graph convolutional neural network to get classification scores. We also analyze the spatial and spectral convolution filtering techniques in graph neural networks. Spectral-based models perform better than spatial-based models and classical CNN with lesser compute cost.

Index Terms— Graph neural network, Superpixels, Quickshift algorithm, Spatial convolution, Spectral convolution

1. INTRODUCTION

Deep learning, particularly convolutional neural networks have in the recent past revolutionized many machine learning tasks. Examples include image classification, video processing, speech recognition, and natural language processing. These applications are usually characterized by data drawn from the Euclidean space. Recently, many studies on extending deep learning approaches for graph data have emerged [1–22]. The motivation for these studies stems from the emergence of applications in which data is drawn from non-euclidean domains and then represented as graphs so as to capture the complex relationships and inter-dependency between objects. Indeed, many datasets and associated problems can be more naturally represented and analyzed as graphs [23]. For instance, graph neural networks (GNNs) have been increasingly used for applications such as molecule and social network classification [24] and generation [25], 3D Mesh classification and correspondence [26], modeling behavior of dynamic interacting objects [27], program synthesis [28], reinforcement learning tasks [29] and many other exciting problems.

While the utility of graph neural networks for emerging applications is promising, the complexity of graph data imposes significant challenges on many existing machine learning algorithms. For instance, in the area of image processing, the use of Graph Convolutional Network (GCN) is still limited to a few examples only [10, 11, 14]. By some carefully hand-crafted graph construction methods or other supervised approaches, images can be converted to structured graphs capable of processing by GCNs. In these GNNs, each pixel of an image is considered as a graph node [15] which is cumbersome and in many cases unnecessary. Instead of learning from individual image pixels, the use of 'superpixels' addresses this concern [30, 31] and helps in reducing the graph size and thereby the computational complexity. Graphs also allow us to impose a relational inductive bias in data for example via prior knowledge. For instance, in case of human pose, the relational bias can be a graph of skeleton joints of a human body [32]. Similarly, in case of videos, relational bias can be a graph of moving bounding boxes [10, 13]. Another example is making inferences about facial attributes and identify by representing facial landmarks as a graph [33]. The literature on application of GNNs on images can be broadly classified in to three groups (a) pixel-based graphs, (b) superpixel-based graphs and (c) object-based graphs - sample illustrations of these three methods are shown in Fig. 1. In addition to providing a comprehensive review of graph techniques for images' superpixels, this paper paper also makes notable contribution by introducing new taxonomy based on how graph represents an image as summarized in Table 1.



(a) (b) (c)

Fig. 1: An illustration of (a) Pixel-based graph, (b) Superpixel-based graph (c) Object-based graph [34].

2. IMAGE BASED GNNS

GNNs on images are characterized by unique challenges with respect to their implementation. Most of the graph neural frameworks [2, 15, 16] are designed for dense representations such as pixel-based graphs. One can achieve effective parallelization on irregular sparse graphs (e.g., superpixel-based graphs) by using fast localized spectral filtering [2] or hierarchichal multi graph spatial convolutional neural networks [31]. Pixel, superpixel and object-based graphs have been extensively used in the literature as summarized in Table 1. For subsequent processing, superpixels have been widely used as an effective way to reduce the number of image primitives. The literature includes numerous methods for determining a superpixel representation from an image, each with different strengths and weaknesses. Recently, many DNN-based methods to

Table 1: Classification of Proposals

Graph Type	Proposals
Pixel-based Graph	[2, 15, 16]
Superpixel-based Graph	[16–18, 30, 31]
Object-based Graph	[3, 19–22, 34]

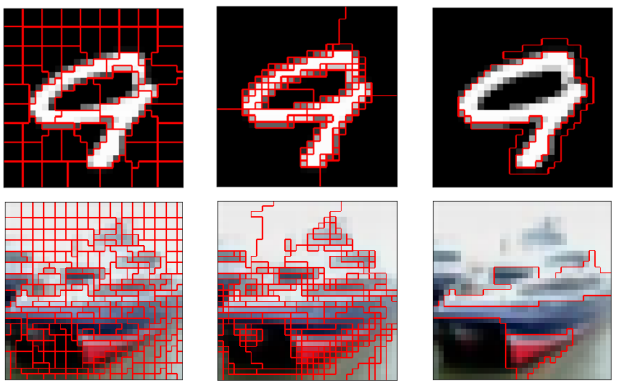


Fig. 2: Superpixel segmentation techniques on MNIST digit: 9 and CIFAR-10 image: Ship.

identify superpixels have been proposed [35, 36]. But the most popular of practices in GNN literature (on account of generally good results and low compute complexity) are SLIC [37], Quickshift [38] and Felzenszwalb [39]. Details of these methods are presented in the following.

2.1. Superpixel segmentation techniques

2.1.1. SLIC

The SLIC (simple linear iterative clustering) [37] algorithm simply performs iterative clustering approach in the 5D space of color information and image location. The algorithm quickly gained momentum and is now widely used due to its speed, storage efficiency, and successful segmentation in terms of color boundaries. However, the limitation of SLIC is that it often captures the background pixels as shown in Fig. 2 – Column 1, and therefore does not significantly help in data reduction for graph generation.

2.1.2. Quickshift

Quickshift [38] is a relatively recent 2D algorithm that is based on an approximation of kernelized mean-shift [40]. It segments an image based on the three parameters: ϵ for the standard deviation of the Gaussian function, α for the weighting the color term and S to limit the calculating a window size of $S \times S$. Therefore, it belongs to the family of local mode-seeking algorithms and is applied to the 5D space consisting of color information and image location. One of the benefits of Quickshift is that it actually computes a hierarchical segmentation on multiple scales simultaneously. As shown in Fig. 2 – Column 2, it does not capture background pixels and also reduces 30% of input data for graph generation.

2.1.3. Felzenszwalb

This fast 2D image segmentation algorithm, proposed in [39], has a single scale parameter that influences the segment size. The actual size and number of segments can vary greatly, depending on local contrast. This segmentation appeared to be less suitable in tests on a series of images, as its parameters requires a special adjustment and consequently a static choice of this parameter leads to unusable results. As shown in Fig. 2 – Column 3, it only captures the pixels corresponding to the region of interest pixels but performs poorly in graph generation procedure.

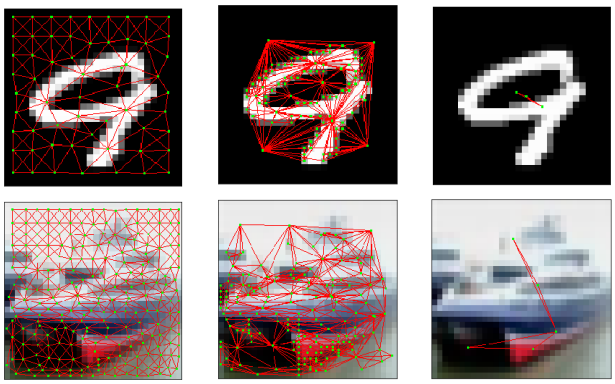


Fig. 3: Region Adjacency Graphs (RAG) generation from SLIC, Quickshift and Felzenszwalb superpixels respectively.

2.2. Graph generation from superpixels

This subsection discusses two methods used for generating graphs from superpixels. These are presented in the following.

2.2.1. Region Adjacency Graph (RAG)

After using a superpixel segmentation technique, a Region Adjacency Graph (RAG) is generated by treating each superpixel as a node and adding edges between all directly adjacent superpixels. Note that this differs from [41] as it only connects to adjacent neighbors as compared to KNN. Each graph node can have associated features, providing an aggregate information based on characteristics of the superpixel itself. The superpixel segmentation technique using Quickshift is one way to divide the image into homogeneous regions. The regions obtained in the segmentation stage are represented as vertices and relations between neighboring regions are represented as edges. The search for the most similar pair of regions is repeated several times per iteration and every search requires $\mathcal{O}(N)$ region similarity computations. The graph is utilized so that the search is limited only to the regions that are directly connected by the graph structure. As we can see in Fig. 3, the representation of an image via a graph G, which consists of superpixels representation S, has far fewer nodes in contrast to the grid representation of the original image. For graph generation, Quickshift (see Fig. 3 – Column 2) performs well as compared to SLIC and Felzenszwalb.

2.2.2. K-Nearest-Neighbors Graph (KNNG)

In a KNN graph, each vertex chooses exactly K nearest neighbors to connect. Superpixel graphs have connections that span more than one neighbour level, with edges formed with the K nearest neighbours [41]. For more details on graph generation, see [41]. For subsequent processing, RAGs have less compute complexity as compared to KNNG. This is the reason for passing RAGs to a graph CNN so as to obtain good classification accuracy.

2.3. Graph Convolutional Neural Network (GCNN)

GCNNs can be broadly classified into two categories: Spectral-based and Spatial-based Models. Spectral-based models define

Table 2: Notations for Algorithms

Variable	Description	Variable	Description
$\mathcal{E}$	Set of edges	$\mathcal{V}$	Set of vertices
n	$\|\mathcal{E}\|$	m	$\|\mathcal{V}\|$
$\mathcal{N}(v)$	Set of neighbors of vertex v	D_G	Diagonal matrix of associated graph (G)
M_G	Region adjacency matrix	W_G	Weighted region adjacency matrix
L_G	Laplacian Matrix	$\{\lambda_l\}_{l=0}^{n-1}$	Frequencies of graph
U	Fourier basis	$\hat{x}$	The graph Fourier transform of signal x
$h_\theta(\Lambda_G)$	Non-parametric filter	L	Number of GNN layers
h_v	Input feature vector of vertex v	h_v^l	Hidden feature vector of vertex v
h_v^L	Output feature vector of vertex v	ϕ_v^l	Node and edge combination feature of layer l

graph convolutions by introducing filters from the perspective of graph signal processing [42]. Spatial-based models on the other hand define convolutions by information propagation. Spatial-based approaches have developed rapidly recently due to their efficiency, flexibility and generality. They are discussed in the following section.

2.3.1. *Spectral Graph Convolutional Neural Networks*

Laplacian is the core operator in spectral graph theory. Laplacian matrix is the most natural form associated with a graph (G). Performing spectral conlvolution on graphs includes four steps as shown in Algorithm 1.

However, there are several issues with this convolutional structure. First, the eigenvector matrix U requires the explicit computation of the eigenvalue decomposition of the graph Laplacian matrix, and hence suffers from the $\mathcal{O}(n^3)$ time complexity which is impractical for large-scale graphs. Second, though the eigenvectors can be precomputed, the time complexity of Algorithm 1 is still $\mathcal{O}(n^3)$. Third, there are $\mathcal{O}(n)$ parameters to be learned in each layer. Besides, these non-parametric filters are not localized in the vertex domain.

To address these limitations, the ChebNet [2] proposed the use of K-polynomial filters in the convolutional layers for localization. To compute fast spectral convolutions in $\mathcal{O}(n)$ time, we have to define compact kernels [2]. Let a non-parametric filter, i.e., a filter whose

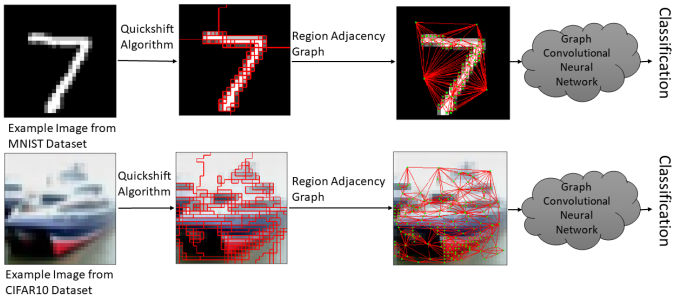


Fig. 4: Illustrative examples of our proposed classification approach

Algorithm 1: Spectral Graph Convolution

Input: x, D_G, M_G
Output: L_G, $\hat{x}$, y

1 Compute graph Laplacian: $L_G \overset{def}{=} D_G - M_G$ Weighted adjacency matrix: $W_G = M_G D_G^{-1}$ Laplacian matrix in normalized form: $L_G = I_n - D_G^{\frac{-1}{2}} W_G D_G^{\frac{-1}{2}}$
2 Compute Fourier functions: $L_G = U\Lambda U^T$
3 Compute Fourier transform: $\hat{x} = U^T x \in \mathbb{R}^n$ and inverse Fourier transform: $x = U\hat{x}$
4 Compute spectral convolution:

$$\begin{aligned} y &= h_\theta(L_G)x \\ &= h_\theta(U\Lambda U^T)x \\ &= U h_\theta(\Lambda_G)U^T x \end{aligned}$$

parameters are all free be defined as

$$h_\theta(\Lambda) = diag(\theta) \tag{1}$$

where parameter $\theta \in \mathbb{R}^n$ is a vector of Fourier coefficients. There are however two limitations with non-parametric filters: (1) They are not localized in space and (2) Their learning complexity is in $\mathcal{O}(n)$, the dimensionality of the data. These issues can be overcome with the use of a polynomial filter [2]:

$$h_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k \Lambda^k, \tag{2}$$

where the parameters $\theta \in \mathbb{R}^K$ is a vector of polynomial coefficients. By [2], (2) can be expressed as Chebyshev polynomial

$$h_\theta(\Lambda) = \sum_k \theta_k T_k(\hat{\Lambda}) \tag{3}$$

of order $K-1$, where the parameter $\theta \in \mathbb{R}^K$ is a vector of Chebyshev coefficients and $T_k(\hat{\Lambda}) \in \mathbb{R}^{n \times n}$ is the Chebyshev polynomial of order K evaluated at $\hat{\Lambda} = \frac{2\Lambda}{\lambda_{max} - I_n}$, a diagonal matrix of scaled eigen values that lie in $[-1, 1]$.

Algorithm 2: Spatial Graph Convolution

Input: Superpixels/Node embeddings
Output: Prediction score

1 Each neighbor sends a message:
$\mathbf{m}_{w,v}^{(l)} = MESSAGE(\mathbf{h}_v^{l-1}, \mathbf{h}_w^{l-1})$
2 Messages are aggregated across all neighbors:
$\mathbf{a}_v^l = AGGREGATE(\mathbf{m}_{w,v}^{(l)} : w \in \mathcal{N}(v))$
3 Neighbor information is used to update node representation:
$\mathbf{h}_v^l = UPDATE(\mathbf{h}_v^{l-1}, \mathbf{a}_v^l)$
4 Prediction based on global readout of node embeddings:
$\phi(G) = MLP(\sum_{v \in V} \mathbf{h}_v^{(L)})$

2.3.2. *Spatial Graph Convolutional Neural Network*

In this subsection, we focus on the spatial graph convolutions that propagate and aggregate the node representations from neighboring

Table 3: Evaluation Results showing comparison between classical CNN as well as state-of-the-art GCNs.

Input	CNN/GCNN Type	Model	Datasets	Accuracy (%)	Compute Cost
Pixel based RAG	Spectral Convolution Filtering	ChebNet [2]	MNIST	99.14	$\mathcal{O}(\log(K\|\mathcal{E}\|))$
		ChebNet [2]	CIFAR-10	98	$\mathcal{O}(\log(K\|E\|))$
Superpixel based RAG	Spatial Graph Filtering	EGCNN [17]	MNIST	98.025	$\mathcal{O}(\log(\|\mathcal{V}^2\|))$
		GCN baseline	MNIST	86.358	$\mathcal{O}(\log(Kmh_i^l + Knh_i^{l^2}))$
		EGCNN [17]	CIFAR-10	75.230	$\mathcal{O}(\log(\mathcal{V}^2))$
		GCN baseline	CIFAR-10	53.606	$\mathcal{O}(\log(Kmh_i^l + Knh_i^{l^2}))$
Image	Classical CNN	VGG-16	MNIST	99.139	-
Image	Classical CNN	VGG-16	CIFAR-10	93.5	-

Table 4: Quickshift Parameters

Dataset	Parameter	N	N_{min}	N_{max}	deg	deg_{min}	deg_{max}
MNIST	$\epsilon = 2, \alpha = 1, S = 2$	82.1	5	154	6.8	1	101
CIFAR-10	$\epsilon = 1, \alpha = 1, S = 5$	182	18	624	7.4	1	67

nodes in the vertex domain [17, 42] as shown in Algorithm 2. The most popular implementation of spatial convolution filtering is given as

$$\mathbf{h}_v^l = \sigma(\mathbf{W}_1\mathbf{h}_v^{(l-1)} + \mathbf{W}_2 \sum_{w \in \mathcal{N}(v)} C_{v,w}\mathbf{h}_w(l-1)) \quad (4)$$

Where W_1, W_2 are trainable parameters and $C_{v,w}$ is Normalization coefficient and structure dependent, i.e., $C_{v,w} = |\mathcal{N}(v)|^{-1}$.

3. EXPERIMENTAL EVALUATION

We evaluate three models: ChebNet [2], EGCNN [17] and GNN baseline on image based graph classification dataset: MNIST and CIFAR-10, in comparison with classical CNN model: VGG-16. ChebNet is spectral-based model while GNN baseline and EGCNN are spatial-based models. For each dataset there is a set of graphs with different number of nodes (superpixels) and each graph G has a single categorical label that is to be predicted. Dataset details, parameters and metrics are discussed in following subsections.

3.1. Datasets

The first dataset is MNIST (Modified National Institute of Standards and Technology database). It is a large database of handwritten digits from 0 to 9. It consists of $55,000$ training images, 5000 validation images and $10,000$ test images. The images are uniform on the scale 28×28 pixels and have only one color channel with grey values. The other dataset is CIFAR-10 (Canadian Institute for Advanced Research dataset). It is a large dataset of $60,000$ color images, each of which is exactly one of 10 classes: airplane, car, bird, cat, deer, dog, frog, horse, ship and truck. It consists of $45,000$ training images, 5000 validation images and $10,000$ test images. The images are uniform size of scale 32×32 pixels and have three color channels. As shown in Fig. 4, 2 preprocessing steps are performed on both of these datasets. The first step is to generate superpixels using Quickshift [38] (see section 2.1.2) and second step is graph generation process from superpixels (see section 2.2.1). Using the Quickshift algorithm, data reduction is approximately 30% of input data i.e. 2150 characteristics of image graph as compared to the number of features of the original image ($32 \times 32 \times 3 = 3072$).

3.2. Parameters and Metrics

Quickshift parameters for MNIST and CIFAR-10 datasets are shown in table 4. We use Softmax Regression 5 for multidimensional classification problem, that is the probability distribution of classes Y for the occurrence of input $\mathbf{x}$. The exponential function in equation 5 highlighted the highest values and weakened the significant values which are below the maximum.

$$softmax(\mathbf{y}) = \frac{exp(\mathbf{y})}{\sum_i^Y exp(\mathbf{y}_i)} \quad (5)$$

The cross-entropy loss is defined over an input set X as

$$H(\mathcal{X}) = \frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} (\hat{y}(\mathbf{x}), log(y(\mathbf{x}))) \quad (6)$$

To evaluate the results of a classification, the accuracy over an input set X as

$$accuracy(\mathcal{X}) = \frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} max(-f + 1, 0) \quad (7)$$

where $f = |argmax(y(\mathbf{x})) - argmax(\hat{y}(\mathbf{x}))|$ and $argmax(y) = n$ if and only if $y_n >= y_m$ for $1 <= m <= Y$. This describes the proportion of correctly classified input data from the respective total amount of training, validation and test data X and is therefore a suitable mean of determining the quality of classification network.

3.3. Quantitative Evaluation

As we can see in table 3, overall spectral-based GNN: ChebNet [2] perform better than the Embedded GCN model and classical model: VGG-16. Although the GCN make use of pixel or superpixel based RAG and classical CNN utilizes original images for classification but the accuracy of ChebNet [2] is better than all other models with lesser compute cost.

4. CONCLUSION

In this survey paper, we thoroughly reviewed and provided several ways for image classification utilizing graph neural networks. Our purposed technique suggests to convert image into superpixels using Quickshift algorithm and then use superpixles to generate region adjacency graph. For classification task, this graph is passed through different GNN models. We also analyzed the spatial and spectral convolutions filtering techniques for classification. Spectral-based models perform better than spatial-based models and classical CNN with lesser compute cost.

5. REFERENCES

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

[2] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in neural information processing systems*, 2016, pp. 3844–3852.

[3] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena, "Structural-rnn: Deep learning on spatio-temporal graphs," in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2016, pp. 5308–5317.

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

[5] Z. Wang, T. Chen, J. Ren, W. Yu, H. Cheng, and L. Lin, "Deep reasoning with knowledge graph for social relationship understanding," *arXiv preprint arXiv:1807.00504*, 2018.

[6] V. G. Satorras and J. B. Estrach, "Few-shot learning with graph neural networks," 2018.

[7] M. Narasimhan, S. Lazebnik, and A. Schwing, "Out of the box: Reasoning with graph convolution nets for factual visual question answering," in *Advances in Neural Information Processing Systems*, 2018, pp. 2654–2665.

[8] H. Hu, J. Gu, Z. Zhang, J. Dai, and Y. Wei, "Relation networks for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3588–3597.

[9] J. Gu, H. Hu, L. Wang, Y. Wei, and J. Dai, "Learning region features for object detection," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 381–395.

[10] X. Wang, Y. Ye, and A. Gupta, "Zero-shot recognition via semantic embeddings and knowledge graphs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6857–6866.

[11] C.-W. Lee, W. Fang, C.-K. Yeh, and Y.-C. Frank Wang, "Multi-label zero-shot learning with structured knowledge graphs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1576–1585.

[12] S. Qi, W. Wang, B. Jia, J. Shen, and S.-C. Zhu, "Learning human-object interactions by graph parsing neural networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 401–417.

[13] K. Marino, R. Salakhutdinov, and A. Gupta, "The more you know: Using knowledge graphs for image classification," *arXiv preprint arXiv:1612.04844*, 2016.

[14] M. Kampffmeyer, Y. Chen, X. Liang, H. Wang, Y. Zhang, and E. P. Xing, "Rethinking knowledge graph propagation for zero-shot learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 487–11 496.

[15] M. Edwards and X. Xie, "Graph based convolutional neural network," *arXiv preprint arXiv:1609.08965*, 2016.

[16] Q. Liu, L. Xiao, J. Yang, and Z. Wei, "Cnn-enhanced graph convolutional network with pixel-and superpixel-level feature fusion for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, 2020.

[17] M. Fey and J. E. Lenssen, "Fast graph representation learning with pytorch geometric," *arXiv preprint arXiv:1903.02428*, 2019.

[18] S. Wan, C. Gong, P. Zhong, B. Du, L. Zhang, and J. Yang, "Multiscale dynamic graph convolutional network for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 58, no. 5, pp. 3162–3177, 2019.

[19] M. Qi, J. Qin, A. Li, Y. Wang, J. Luo, and L. Van Gool, "stagnet: An attentive semantic rnn for group activity recognition," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 101–117.

[20] P. Zhou and M. Chi, "Relation parsing neural network for human-object interaction detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 843–851.

[21] F. Z. Zhang, D. Campbell, and S. Gould, "Spatio-attentive graphs for human-object interaction detection," *arXiv preprint arXiv:2012.06060*, 2020.

[22] J. Tompson, A. Jain, Y. LeCun, and C. Bregler, "Joint training of a convolutional network and a graphical model for human pose estimation," *arXiv preprint arXiv:1406.2984*, 2014.

[23] N. T. Bliss and M. C. Schmidt, "Confronting the challenges of graphs and networks," *Lincoln laboratory journal*, vol. 20, no. 1, 2013.

[24] B. Knyazev, X. Lin, and R. Mohamed, "Amer, and graham w taylor. spectral multigraph networks for discovering and fusing relationships in molecules," in *NeurIPS Workshop on Machine Learning for Molecules and Materials*, 2018.

[25] M. Simonovsky and N. Komodakis, "Dynamic edge-conditioned filters in convolutional neural networks on graphs," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3693–3702.

[26] M. Fey, J. E. Lenssen, F. Weichert, and H. Müller, "Splinecnn: Fast geometric deep learning with continuous b-spline kernels," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 869–877.

[27] T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, and R. Zemel, "Neural relational inference for interacting systems," in *International Conference on Machine Learning*. PMLR, 2018, pp. 2688–2697.

[28] M. Allamanis, M. Brockschmidt, and M. Khademi, "Learning to represent programs with graphs," *arXiv preprint arXiv:1711.00740*, 2017.

[29] V. Bapst, A. Sanchez-Gonzalez, C. Doersch, K. Stachenfeld, P. Kohli, P. Battaglia, and J. Hamrick, "Structured agents for physical construction," in *International Conference on Machine Learning*. PMLR, 2019, pp. 464–474.

[30] X. Liang, X. Shen, J. Feng, L. Lin, and S. Yan, "Semantic object parsing with graph lstm," in *European Conference on Computer Vision*. Springer, 2016, pp. 125–143.

[31] B. Knyazev, X. Lin, M. R. Amer, and G. W. Taylor, "Image classification with hierarchical multigraph networks," *arXiv preprint arXiv:1907.09000*, 2019.

[32] S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[33] E. Antonakos, J. Alabort-i Medina, and S. Zafeiriou, "Active pictorial structures," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5435–5444.

[34] Y. Jiang, H. Koppula, and A. Saxena, "Hallucinated humans as the hidden context for labeling 3d scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2993–3000.

[35] F. Yang, Q. Sun, H. Jin, and Z. Zhou, "Superpixel segmentation with fully convolutional networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 13 964–13 973.

[36] V. Jampani, D. Sun, M.-Y. Liu, M.-H. Yang, and J. Kautz, "Superpixel sampling networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 352–368.

[37] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012.

[38] A. Vedaldi and S. Soatto, "Quick shift and kernel methods for mode seeking," in *European conference on computer vision*. Springer, 2008, pp. 705–718.

[39] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *International journal of computer vision*, vol. 59, no. 2, pp. 167–181, 2004.

[40] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 5, pp. 603–619, 2002.

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

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