Image Classification using Graph-based Representations and Graph Neural Networks

Giannis Nikolentzos[1], Michalis Thomas[1], Adín Ramírez Rivera[3], and Michalis Vazirgiannis[1,2]

[1] Athens University of Economics and Business, Greece
{nikolentzos,p3150048,mvazirg}@aueb.gr
[2] École Poytechnique, France
[3] University of Campinas, Brazil
adin@ic.unicamp.br

Abstract. Image classification is an important, real-world problem that arises in many contexts. To date, convolutional neural networks (CNNs) are the state-of-the-art deep learning method for image classification since these models are naturally suited to problems where the coordinates of the underlying data representation have a grid structure. On the other hand, in recent years, there is a growing interest in mapping data from different domains to graph structures. Such approaches proved to be quite successful in different domains including physics, chemoinformatics and natural language processing. In this paper, we propose to represent images as graphs and capitalize on well-established neural network architectures developed for graph-structured data to deal with image-related tasks. The proposed models are evaluated experimentally in image classification tasks, and are compared with standard CNN architectures. Results show that the proposed models are very competitive, and yield in most cases accuracies better or comparable to those of the CNNs.

Keywords: graph-based representations, graph neural networks, image classification

1 Introduction

Image classification is a fundamental task in computer vision, where the goal is to classify an image based on its visual content. For instance, we can train an image classification algorithm to answer if a car is present in an image or not. While detecting an object is trivial for humans, robust image classification is still a challenge in computer vision applications.

In the past years, convolutional neural network architectures (CNNs) have proven extremely successful on a wide variety of tasks in computer vision [14]. These models are naturally suited to problems where the input data take the form of a regular grid, and exhibit some inherent statistical properties such as

Pre-print. Published in Int. Conf. of Complex Networks

local stationarity and compositionality. Images are examples of data that fall into this category.

In many domains, data is commonly represented as graphs. This is mainly due to the rich representation capabilities that these structures exhibit. Graphs can model both the entities and the relationships between them. Typically, the vertices of a graph correspond to some entities, and the edges model how these entities interact with each other. In a collaboration network, such interactions may for instance correspond to collaborations in a network of scientists. Graphs are a very flexible means of data representation, and several fundamental data structures can be thought of as instances of graphs. For example, a sequence can be thought of as a graph, with one node per element and edges between consecutive elements. In some cases, even data that does not exhibit graph-like structure like text is mapped to graph representations [20]. In the past years, a vast number of learning algorithms has been developed in order to work with graphs and process the information they represent. There are now available neural network models which have achieved state-of-the-art performance on many real-world graph classification datasets [23]. More specifically, an explosion in research activity in the field of graph neural networks has taken place in the last few years.

In this paper, we propose to represent images as graphs, and to apply machine learning algorithms that operate on graphs to the emerging representations. Specifically, we present different approaches for representing images as graphs, and we capitalize on well-established neural network architectures developed for graph-structured data to deal with image classification tasks. The proposed models exploit properties inherent in images such as stationarity of statistics and locality of pixel dependencies. We evaluate the proposed models in image classification tasks, and we compare them with standard CNN architectures. We also study the robustness of the proposed models to transformations of the input images and to adversarial attacks. Results show that the proposed models are very competitive, and yield in most cases accuracies better or comparable to those of the CNNs. It should be mentioned that this is not the first work to apply graph neural networks to image data. However, in this paper, we evaluate a large combination of representations and graph neural network architectures, and to the best of our knowledge, this is the most complete evaluation to date of graph representations in computer vision, and graph neural networks for image-related tasks.

2 Related Work

Graph-based representations of images have a long history in the field of pattern recognition. A detailed review of these approaches is beyond the scope of this paper; we refer the interested reader to the work of Conte *et al.* [5] and Vento and Foggia [22].

The problem of image classification has been widely studied over the past years, while several of the proposed approaches borrowed ideas from graph min-

ing techniques. For instance, some works have proposed the use of graph kernels and have studied their effectiveness in image classification [4, 7, 12], while others have produced new image representations using graph-based features [1, 26]. Graph neural networks have been recently applied to image classification tasks. Specifically, some recent graph neural network models were evaluated on the benchmark MNIST classification problem [3, 6, 19]. The main difference between these works and ours is that they follow different approaches for representing images as graphs. For instance, Defferard *et al.* construct in [6] a weighted k-NN similarity graph where nodes represent pixels and each pixel is connected to its k most similar pixels in terms of intensity. The weights of the edges are computed using a function similar to the radial basis function kernel. Simonovsky and Komodakis represented in [19] each image as a point cloud with coordinates $(x, y, 0)$ where $x, y \in \{0, \ldots, 27\}$, while Bruna *et al.* subsample the normal 28×28 grid to get 400 coordinates [3]. Furthermore, all these works apply a single architecture to the MNIST dataset, while in our work, we evaluate a series of message passing layers and readout functions. Very recently, graph neural networks have been also applied to other computer vision tasks, such as to the problem of image matching (i.e., to find correspondences between points in images) [18, 25].

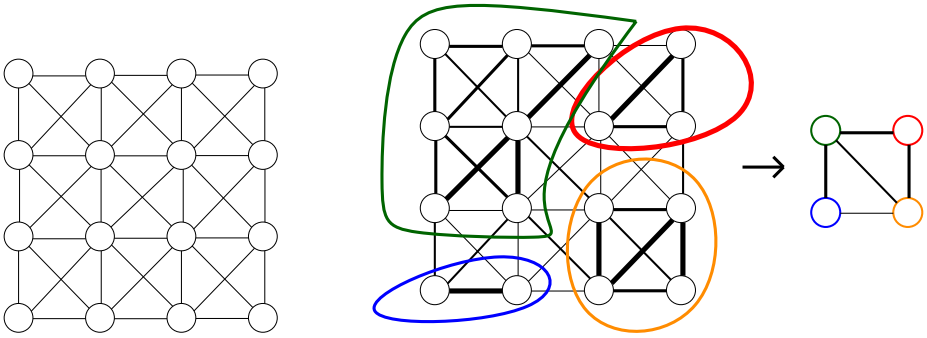
3 Models and Representations

In this section, we present the graph representations of images that we employed and the different models that we applied to these representations. We start by fixing our notation. Let $G = (V, E)$ be an undirected graph consisting of a set V of nodes and a set E of edges between them. We will denote by n the number of nodes and by m the number of edges. The neighborhood of a node $v \in V$ is the set of all vertices adjacent to v, that is $\mathcal{N}(v) = \{u : (v, u) \in E\}$ where (v, u) is an edge between vertices v and u. The graph representations that we utilize are node-attributed graphs. That is, each node is annotated with one or more attributes.

3.1 Graph-based Representations of Images

In the past, several approaches have been proposed for mapping images to graph structures. Almost all existing approaches are ad-hoc and are generally motivated by performance considerations. One usually adopts the representation that is shown to perform best in the considered task. In this paper, we experiment with two different graph representations of images, namely the king's graph and a coarsened graph which we derive from the output of some community detection algorithm. We illustrate the two considered graph representations in Figure 1.

King's graph. The $m \times n$ king's graph is a graph with mn vertices in which each vertex represents a square in an $m \times n$ chessboard, and each edge corresponds to a legal move by a king. The $m \times n$ king's graph can be constructed as the strong product of the path graphs P_m and P_n. In other words, the king's graph



(a) king's graph (b) coarsened graph

Fig. 1. The two considered graph representations of images: (a) king's graph, and (b) a coarsened graph whose nodes correspond to communities extracted from a weighted variant of the king's graph.

is a graph whose nodes (except those belonging to the border of the grid) are connected with their 8 neighborhood nodes by an edge. In our setting, each node of the graph represents a pixel. Furthermore, each node is annotated with a real value (i.e., intensity of the pixel) in case of grayscale images or a 3-dimensional vector (i.e., intensity of the RGB channels) in the case of colored images.

Coarsened graph. In the aforementioned representation, each node corresponds to a pixel in the input image. Since the number of pixels is usually large (even for low resolution images), we propose to use community detection algorithms to reduce the number of nodes in the graph to a representative subsample of pixels or regions. We start from the aforementioned king's graph representation and we transform it into a weighted graph where edge weights capture the similarity between pairs of pixels. We assume that there is no a priori knowledge about the components of the input image, and therefore, we use the following function to compute the weight of the edge between two vertices v_i and v_j:

$$w_{ij} = 1 - \|\mathbf{x}_i - \mathbf{x}_j\|. \tag{1}$$

Here, we have assumed that the pixel intensities take values between 0 and 1 and, therefore, $0 \leq w_{ij} \leq 1$ holds. Other functions such as the Gaussian kernel could also be employed. To extract the representative subsample of pixels, we apply the Louvain method, a well-known community detection algorithm [2]. The algorithm returns a set of communities, and we treat each community as a node in the new graph. In the new graph, two nodes (i.e., communities) are linked to each other by an edge if one or more pixels of the one community was connected to one or more pixels of the second community in the original king's graph. Furthermore, each node is annotated with the average of the intensities (or vectors in case of colored images) of the pixels that belong to the corresponding community.

3.2 Models

Graph neural networks (GNNs) have attracted a lot of attention in the past years. Most GNNs share the same basic idea, and can be reformulated into a single common framework [9]. A GNN model consists of a series of *message passing* (MP) layers. Each one of these layers uses the graph structure and the node feature vectors from the previous layer to generate new representations for the nodes. The feature vectors are updated by aggregating local neighborhood information. To generate a vector representation over the whole graph, GNNs apply a *readout* function to node representations generated by the final message passing layer. We next present three general models which we will evaluate in image classification. Note that two of the models (MP+CNN and MP+Pool+Readout) are specifically designed for graph representations of images that exhibit a grid-like structure, and cannot be applied to general graphs.

MP+Readout. This model consists of a series of message passing layers followed by a readout function. Each message passing layer updates the representation of each node based on the representations of its neighbors and possibly on the node's own previous representation. Then, to produce an image representation, the model applies a readout function to the node representations of the final message passing layer.

MP+CNN. This model consists of a series of message passing layers followed by a CNN. The message passing procedure can be seen as a method for updating the features of the pixels. Therefore, after T message passing layers, we end up with an "image" with as many channels as the hidden dimension of the final message passing layer. This "image" can be passed on to a standard CNN model to produce a representation for the input image. Note that this model is specifically designed for the image classification task where images are modeled as king's graphs, and it cannot be applied to general graphs.

MP+Pool+Readout. This model consists of a series of message passing and pooling layers followed by a readout function. A pooling layer is in fact a clustering layer which replaces a set of nodes with a single node. We incorporate spatial relations between pixels into clustering, and thus pixels are clustered together with their neighbors. Clearly, this model can only be applied to images represented as king's graphs. Each pooling layer halves the size of each dimension of the grid. The features of the new nodes are computed using some permutation invariant function on the cluster's nodes (e.g., sum, mean or max). Therefore, this model consists of alternating convolutional and pooling (i.e., clustering) layers in the same spirit as CNNs are composed of alternating convolutional and pooling layers. To produce an image representation, the model applies a readout function to the node representations of the final pooling layer.

Employed message passing layers. In this work, we experimented with the following five message passing layers: Gated Graph layer [15], GCN [13],

GAT [21], GraphSAGE [11], and 1-GNN [16]. Due to space constraints, we cannot provide more details about the different message passing layers. For a detailed description of each message passing layer, we refer the reader to their respective papers.

Employed readout functions. We utilized the following four readout functions: (1) *sum*: it computes the sum of the node representations; (2) *mean*: it computes the average of the node representations; (3) *max*: this operator computes a vector representation for the graph where each element is equal to the maximum value of the corresponding elements of all node representations; and (4) *SortPool* [24]: this layer generates graph representations of specific size by first sorting the nodes of the graph, and then retaining only the first k nodes. If the number of nodes is less than k, zero-padding is applied. To rank the nodes, the SortPool layer sorts the last element of the nodes' representations in a descending order.

4 Experimental Evaluation

We perform all our experiments on the MNIST dataset of handwritten digits. The dataset is split into a training set and a test set of 60 000 and 10 000 images, respectively. Each image is a 28×28 pixel square. There are 10 digits in total (from 0 to 9), and therefore 10 different classes. We represent each image as a 28×28 king's graph as discussed above. All the images share the same underlying graph structure, however, their nodes are annotated possibly with different attributes. We assign weights to the edges of the king's graph using the function shown in (1), and then we apply the Louvain algorithm to obtain the set of communities and generate the coarsened graph. Note that the Louvain graph automatically detects the number of communities. Hence, the coarsened graph representations of some images may have different number of nodes than others (we found that the number of nodes of the emerging graphs ranges from 10 to 20).

4.1 Model Selection

We created a training and a validation set of images (the two sets are disjoint) by randomly sampling 10 000 and 1000 images from the training set of MNIST, respectively. We trained the models on the 10 000 images and report their accuracy on the 1000 images of the validation set. The representations of the images produced by the different models are passed on to a 2-layer multilayer perceptron (MLP) with a softmax activation function in the output. For all configurations, we train the neural networks for 100 epochs. We use the Adam optimizer with a learning rate of 0.001. The batch size is set equal to 64. All our dense layers use ReLU activation. To prevent over-fitting, we use dropout with a rate of 0.2. The hyper-parameters we tune are: (1) the number of message passing layers $\in \{2, 3, 4\}$ for MP+Readout and MP+CNN, and $\in \{2, 3\}$ for

Table 1. Performance of the different combinations of graph representations, message passing layers and readout functions on the validation set of the MNIST dataset.

	MP layer	MP+Readout				MP+CNN	MP+Pool+Readout			
		Sum	Max	Mean	SortPool		Sum	Max	Mean	SortPool
king's graph	GAT	80.2	73.8	60.9	41.3	96.6	93.3	91.3	91.0	94.6
	GCN	76.4	66.8	52.0	32.7	96.4	93.3	92.8	93.3	93.2
	GraphSAGE	79.5	54.6	56.9	33.0	97.1	92.6	91.9	93.3	90.9
	1-GNN	95.5	94.5	95.4	63.8	97.1	97.6	97.8	97.5	96.7
	Gated Graph layer	96.9	95.1	95.6	67.2	96.4	97.3	97.9	96.7	97.1
coars. graph	GAT	65.9	66.5	65.0	49.8	–	–	–	–	–
	GCN	61.4	60.9	61.4	46.3	–	–	–	–	–
	GraphSAGE	59.0	59.1	55.6	42.8	–	–	–	–	–
	1-GNN	75.1	75.2	75.4	66.2	–	–	–	–	–
	Gated Graph layer	78.3	78.5	73.2	67.7	–	–	–	–	–

MP+Pool+Readout; (2) the number of hidden units of the message passing layers $\in \{16, 64, 128\}$ for MP+Readout and MP+Pool+Readout and $\in \{4, 16, 32\}$ for GNN +CNN; (3) the number of hidden units of the MLP layer $\in \{128, 256\}$ for all models. For MP+Pool+Readout, we also tune the number of sub-sampling (i.e., clustering) layers $\in \{2, 3, 4\}$, and the type of the aggregation function of the features of the clustered nodes $\in \{\text{Sum}, \text{Mean}, \text{Max}\}$. For the SortPool readout function, k was set equal to 20. The baseline CNN and the CNN component of MP+CNN consist of two convolutional layers. The first layer contains 16 filters of size 4×4, while the second layer contains 32 filters of size 3×3. Both convolutional layers are followed by max-pooling layers of size 2×2.

Table 1 illustrates the classification accuracies obtained from the different models. Note that in the case of the coarsened graphs, we can only apply the MP+Readout models. Indeed, the MP+CNN and MP+Pool+Readout models cannot be applied since the input data does not take the form of a regular grid anymore, and moreover, the graph has already been clustered.

We first focus on the MP+Readout model. We find that the king's graph representation yields higher accuracies compared to the coarsened graph representation. In all cases, the difference in performance is significant. We believe that this is due to the information loss associated with the coarsening procedure (groups of nodes and their features are merged together). With regards to the different message passing layers, our results indicate that the 1-GNN and Gated Graph layer achieve much higher accuracies than the rest of the layers. Interestingly, these are the two layers that do not use mean aggregators in the message passing procedure. Mean aggregators capture the distribution of the features in the neighborhood of a node. Thus, they may fail to distinguish the exact neighbors of a node. We next compare the different readout functions to each other. We can see that SortPool is the worst-performing function. We believe that this is due to the fact that it ignores a large number of node representations. In the

Table 2. Performance of the different message passing layers of the MP+CNN model on the validation set of the MNIST dataset for undirected and directed king's graph representations.

MP layer	Undirected	Directed
GAT	96.6	96.7
GCN	96.4	96.3
SAGE	97.1	96.8
1-GNN	97.1	97.3
Gated Graph layer	96.4	98.0

Table 3. Performance of the selected models on the full MNIST dataset.

Model	MNIST
CNN	99.1
MP+Readout	96.4
MP+Readout (coarsened graph)	72.7
MP+CNN	**99.4**
MP+Pool+Readout	98.9

case of the king's graph representation, Sum reached the highest accuracies, while Max and Mean reached the second and the third best accuracy levels among all considered functions. On the other hand, in the case of coarsened graphs, Max is the best-performing function. The Sum function yielded similarly good results, while Mean produced slightly worse results than the other two functions.

As discussed above, the MP+CNN model can only be applied to the king's graph representation of the images. This model delivers the "best of both worlds" from GNNs and CNNs. It combines the representational capacity of GNNs with the ability of CNNs to effectively deal with image data. In Table 1, we can see that it also achieves very high accuracies, regardless of the employed message passing layer. Interestingly, message passing layers that failed to achieve high levels of accuracy when integrated into the MP+Readout model (e.g., GAT, GCN, GraphSAGE), now achieve accuracies close to the maximum observed.

The MP+Pool+Readout model is also applied only to the king's graph representation of images. Clearly, the MP+Pool+Readout model improves over the MP+Readout model for all combinations of message passing layers and readout functions. This highlights that clustering neighboring pixels is highly beneficial when dealing with image data due to the statistical properties inherent to this kind of data such as local stationarity and compositionality. It should be mentioned that one of the variants of the MP+Pool+Readout model (the one that uses the Gated Graph message passing layer and the Max readout function) achieved the highest validation accuracy among all considered models.

We also studied the impact of edge directions on the performance of the models. We assign directions to all the edges such that they start from nodes on the top and/or left and end at nodes on the bottom and/or right. We use the MP+CNN model to evaluate these representations. Table 2 illustrates the obtained accuracies for different message passing layers. We can see that for almost all layers, the use of directed edges has almost no impact on the classification accuracy. Notably, in the case of the Gated Graph layer, the use of the directed king's graph representation led to an absolute improvement of 1.6%.

4.2 Image Classification

We next evaluate the architectures that performed best in our model selection experiments on the full MNIST dataset. The obtained accuracies are shown in Table 3. We can see that MP+CNN achieves the highest accuracy, followed by CNN, MP+Pool+Readout, MP+Readout and MP+Readout (applied to the coarsened graph) in that order. Excluding the model applied to the coarsened graph, the difference in performance between the rest of the models is relatively small, indicating that all of them are effective in classifying the images contained in the MNIST dataset.

4.3 Robustness to Affine Transformations

We next investigate the sensitivity of the proposed models against affine transformations applied to the images of MNIST. Such natural transformations can be used to completely fool image classification models. We apply three different types of transformations: scaling, rotation and translation. In all three cases, we experimented with the full MNIST dataset, i.e., 60 000 training samples and 10 000 test samples. We next present the experimental setup for each one of the three transformations.

- *Scaling*: Given a scaling factor k, we scale down all the images of the test set as follows: for each image, we randomly sample a scaling factor from $1/10$ to $1/k$ by steps of $1/10$, i.e., we randomly sample one of the elements of the set $\{1/10, 2/10, 3/10, \ldots, 1/k\}$ with equal probability. We also scale down 5% of the images of the training set following the same procedure.
- *Rotation*: In order to ensure that the perturbed images are not heavily distorted, we restrict our rotations to a maximum of 20°. Specifically, we run a series of experiments where in each experiment, we rotate all the images of the test set by a specific amount of degree, while we rotate no images of the training set.
- *Translation*: The translation is applied as follows. We first randomly sample a valid pair of cardinal directions (i.e., northeast, southeast, southwest, and northwest) with uniform probability (i.e., 0.25 for each pair). Then, the image is shifted by no more than k pixels along each one of the two directions. For each direction, the amount of shift is chosen with uniform probability from $\{0, 1, \ldots, k\}$. This transformation is applied to all the images of the test set and to 5% of the training data.

We show the results for the three types of transformations in Figure 2. Interestingly, scaling and rotation do not have such a large impact on the performance of the models as translation. We observe that MP+CNN is generally the most robust approach to the three types of transformations. Specifically, it outperforms the other approaches across all considered scaling factors and rotation degrees. It is only outperformed by the other approaches in the case of largely-translated images (maximum translation greater than 7 pixels).

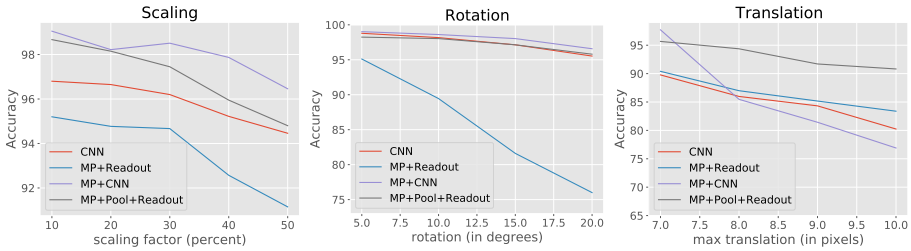


Fig. 2. Performance of the different models under transformations of test set.

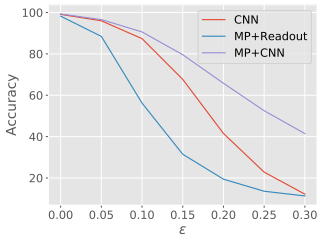


Fig. 3. Performance of the different models with respect to the amount of perturbation applied to the images of the test set.

MP+Cluster+Readout achieves the second best accuracy levels among all models considered. It outperforms CNN and MP+Readout in the case of scaled and translated images, while it yields similar results to CNN in the case of rotated images. MP+Readout seems to be more invariant to scale-transformed images than CNN since it achieves higher accuracies in this set of experiments. However, the same does not hold for images that have undergone rotation or translation. In the latter case, MP+Readout is outperformed by all the other models by very wide margins.

4.4 Robustness Against Adversarial Examples

It has been shown that many classes of machine learning algorithms are vulnerable to adversarial manipulation of their input. This can often lead to incorrect classification. More specifically, neural networks are highly vulnerable to attacks based on small modifications of the input to the model at test time [10, 17]. An interesting research direction is to investigate how robust the different models are against adversarial attacks. To this end, we follow the work presented by [10] and we create a set of adversarial samples by applying perturbations to the images of the test set as follows: $\tilde{\mathbf{X}} = \mathbf{X} + \mathbf{P}$ where $\mathbf{X}$ is an image and $\mathbf{P}$ the matrix that contains the perturbations for the different elements of the image.

It turns out that if worst-case perturbations are applied to test samples, then a model might produce incorrect predictions with high confidence [10]. We next present how such worst-case perturbations can be produced. Let $\boldsymbol{\theta}$ be the

parameters of a model, $\mathbf{X}$ the input to the model, y the target associated with $\mathbf{X}$ and $J(\boldsymbol{\theta}, \mathbf{X}, y)$ be the loss function used to train the neural network. An optimal max-norm constrained perturbation can be obtained as follows:

$$\mathbf{P} = \epsilon \operatorname{sign}(\nabla_{\mathbf{X}} J(\boldsymbol{\theta}, \mathbf{X}, y)) \tag{2}$$

We use the above approach to produce 10 000 adversarial samples from the test images of MNIST. We use different values of ϵ that range from 0 to 0.3. We train the models on the standard training set of MNIST (i.e., 60 000 samples), and then we evaluate them on the set of adversarial samples. Figure 3 illustrates the obtained results. We observe that the performance of the different models decreases significantly as the value of ϵ increases. This is not surprising since the greater the value of ϵ, the larger the amount of perturbation that is applied to the images of the test set. Clearly, MP+CNN outperforms the other models for the different values of ϵ. CNN produced the second best results. Note that CNN and MP+Readout yield an error rate greater than 87% when ϵ is equal to 0.3.

5 Conclusion

In this paper, we proposed to represent images as graphs, and then to employ graph neural networks to deal with image-related learning tasks such as image classification. We evaluated several combinations of graph representations and graph neural network architectures, and found that the proposed models are very competitive, and achieve accuracies comparable to those of CNNs.

Acknowledgement

This research is co-financed by Greece and the European Union (European Social Fund- ESF) through the Operational Programme «Human Resources Development, Education and Lifelong Learning»in the context of the project "Reinforcement of Postdoctoral Researchers - 2nd Cycle" (MIS-5033021), implemented by the State Scholarships Foundation (IKY).

References

1. Niusvel Acosta-Mendoza, Andrés Gago-Alonso, and José E Medina-Pagola. Frequent approximate subgraphs as features for graph-based image classification. *Knowledge-Based Systems*, 27:381–392, 2012.
2. Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.
3. Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and deep locally connected networks on graphs. In *2nd International Conference on Learning Representations*, 2014.

4. Gustavo Camps-Valls, Nino Shervashidze, and Karsten M Borgwardt. Spatio-spectral remote sensing image classification with graph kernels. *IEEE Geoscience and Remote Sensing Letters*, 7(4):741–745, 2010.
5. Donatello Conte, Pasquale Foggia, Carlo Sansone, and Mario Vento. How and why pattern recognition and computer vision applications use graph. In *Applied Graph Theory in Computer Vision and Pattern Recognition*, pages 85–135. 2007.
6. Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852, 2016.
7. Olivier Duchenne, Armand Joulin, and Jean Ponce. A graph-matching kernel for object categorization. In *Proceedings of the 2011 International Conference on Computer Vision*, pages 1792–1799, 2011.
8. Matthias Fey, Jan E Lenssen, Christopher Morris, Jonathan Masci, and Nils M Kriege. Deep graph matching consensus. In *8th International Conference on Learning Representations*, 2020.
9. Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1263–1272, 2017.
10. Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *3rd International Conference on Learning Representations*, 2015.
11. Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.
12. Zaïd Harchaoui and Francis Bach. Image classification with segmentation graph kernels. In *Proceedings of the 2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.
13. Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.
14. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.
15. Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.
16. Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and Leman go neural: Higher-order graph neural networks. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, pages 4602–4609, 2019.
17. Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *Proceedings of the 1st IEEE European Symposium on Security and Privacy*, pages 372–387, 2016.
18. Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperGlue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4938–4947, 2020.
19. Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3693–3702, 2017.

20. Michalis Vazirgiannis, Fragkiskos D Malliaros, and Giannis Nikolentzos. GraphRep: Boosting text mining, NLP and information retrieval with graphs. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 2295–2296, 2018.
21. Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.
22. Mario Vento and Pasquale Foggia. Graph matching techniques for computer vision. In *Image Processing: Concepts, Methodologies, Tools, and Applications*, pages 381–421. 2013.
23. Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*, 2019.
24. Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *In Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages 4438–4445, 2018.
25. Zhen Zhang and Wee Sun Lee. Deep graphical feature learning for the feature matching problem. In *Proceedings of the 2019 IEEE International Conference on Computer Vision*, pages 5087–5096, 2019.
26. Miao Zheng, Jiajun Bu, Chun Chen, Can Wang, Lijun Zhang, Guang Qiu, and Deng Cai. Graph regularized sparse coding for image representation. *IEEE Transactions on Image Processing*, 20(5):1327–1336, 2010.