Image Classification with Hierarchical Multigraph Networks

Boris Knyazev* [1,2]
bknyazev@uoguelph.ca
Xiao Lin[3]
xiao.lin@sri.com
Mohamed R. Amer* [4]
mohamed@robust.ai
Graham W. Taylor[1,2,5]
gwtaylor@uoguelph.ca

[1] University of Guelph
Canada
[2] Vector Institute for Artificial Intelligence
Canada
[3] SRI International
Princeton, NJ, USA
[4] Robust.AI
[5] Canada CIFAR AI Chair

Abstract

Graph Convolutional Networks (GCNs) are a class of general models that can learn from graph structured data. Despite being general, GCNs are admittedly inferior to convolutional neural networks (CNNs) when applied to vision tasks, mainly due to the lack of domain knowledge that is hardcoded into CNNs, such as spatially oriented translation invariant filters. However, a great advantage of GCNs is the ability to work on irregular inputs, such as superpixels of images. This could significantly reduce the computational cost of image reasoning tasks. Another key advantage inherent to GCNs is the natural ability to model multirelational data. Building upon these two promising properties, in this work, we show best practices for designing GCNs for image classification; in some cases even outperforming CNNs on the MNIST, CIFAR-10 and PASCAL image datasets.

1 Introduction

In image recognition, input data fed to models tend to be high dimensional. Even for tiny MNIST [23] images, the input is $28 \times 28\text{px} = 784$ dimensional and for larger PASCAL [10] images it explodes to $333 \times 500\text{px} \approx 165,000$ dimensions (Figure 1). Learning from such a high-dimensional input is challenging and requires a lot of labelled data and regularization. Convolutional Neural Networks (CNNs) successfully address these challenges by exploiting the properties of shift-invariance, locality and compositionality of images [5]. We consider an alternative approach and instead reduce the input dimensionality. One simple way to achieve that is downsampling. But in this case, we may lose vital structural information, so to better preserve it, we extract superpixels [1, 24]. Representing a set of superpixels such that CNNs could digest and learn from them is non-trivial. To that end, we adopt a strategy from chemistry, physics and social networks, where structured data are expressed by graphs [3, 5, 14]. By defining operations on graphs analogous to spectral [6] or spatial [19] convolution, Graph Convolutional Networks (GCNs) extend CNNs to graph-based data, and show successful applications in graph/node classification [11, 26, 31, 35] and link prediction [29].

*Most of this work was done while the authors were at SRI International.

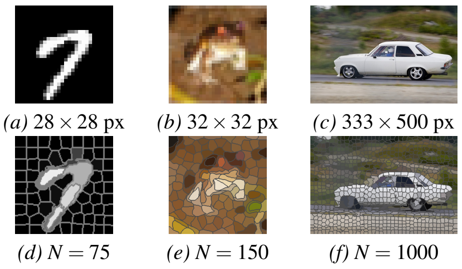


Figure 1: Examples of the original images *(a-c)*, defined on a regular grid, and their superpixel representations *(d-f)* for MNIST *(a,d)*, CIFAR-10 *(b,e)* and PASCAL *(c,f)*; N is the number of superpixels (nodes in our graphs). GCNs can learn both from images and superpixels due to their flexibility, whereas standard CNNs can learn only from images defined on a regular grid *(a-c)*.

The challenge of generalizing convolution to graphs is to have anisotropic filters (such as edge detectors). Anisotropic models, such as MoNet [26] and SplineCNN [11], rely on coordinate structure, work well for various vision tasks, but are often too computationally expensive and suboptimal for graph problems, in which the coordinates are not well defined [20]. While these and other general models exist [3, 13], we rely on widely used graph convolutional networks (GCNs) [19] and their multiscale extension, Chebyshev GCNs (ChebyNets) [8] that enjoy an explicit control of receptive field size.

Our focus is on multigraphs, those graphs that are permitted to have multiple edges, *e.g.* two objects can be connected by edges of different types [25]: (`is_part_of`, `action`, `distance`, etc.). Multigraphs enable us to model spatial and hierarchical structure inherent to images. By exploiting multiple relationships, we can capture global patterns in input graphs, which is hard to achieve with a single relation, because most GCNs aggregate information in a small local neighbourhood and simply increasing its size, as in ChebyNet, can quickly make the representation too entangled (due to averaging over too many features). Hence, methods such as Deep Graph Infomax [36] were proposed. Using multigraph networks is another approach to increase the receptive field size and disentangle the representation in a principled way, which we show to be promising.

In this work, we model images as sets of superpixels (Figure 1) and formulate image classification as a multigraph classification problem (Section 4). We first overview graph convolution (Section 2) and then adopt and extend relation type fusion methods from [20] to improve the expressive power of our GCNs (Section 3). To improve classification accuracy, we represent an image as a multigraph and propose learnable (Section 4.1.1) and hierarchical (Section 4.1.2) relation types that we fuse to obtain a final rich representation of an image. On a number of experiments on the MNIST, CIFAR-10, and PASCAL image datasets, we evaluate our model and show a significant increase in accuracy, outperforming CNNs in some tasks (Section 5).

2 Graph Convolution

We consider undirected graphs $\mathcal{G} = (\mathcal{V}, A)$ with N nodes $\mathcal{V}$ and edges with values in the range $[0,1]$ represented as an adjacency matrix $A \in \mathbb{R}^{N \times N}$. Nodes $v_i \in \mathcal{V}$ usually represent specific semantic concepts such as objects in images [28]. Nodes can also denote abstract blocks of information with common properties, such as superpixels in our case. Edges A_{ij} $(i, j \in [1,N])$ define the relationships and scope of which node effects propagate.

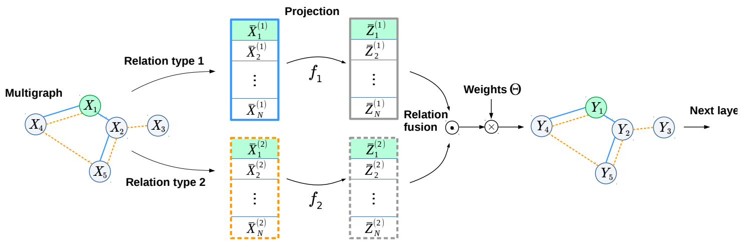


Figure 2: An example of the "PC" relation type fusion based on a trainable projection (Eq. 3). We first project features onto a common multirelational space, where we fuse them using a fusion operator, such as summation or concatenation. In this work, relation types 1 and 2 can denote spatial and hierarchical (or learned) edges. We also allow for three or more relation types.

Convolution is an essential computational block in graph networks, since it permits the gradual aggregation of information from neighbouring nodes. Following [19], for some C-dimensional features over nodes $X \in \mathbb{R}^{N \times C}$ and trainable filters $\Theta \in \mathbb{R}^{C \times F}$, convolution on a graph $\mathcal{G}$ can be defined as:

$$Y = \bar{X}\Theta, \tag{1}$$

where $\bar{X} = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}X$ are features averaged over one-hop ($K = 1$) neighbourhoods, $\tilde{A} = A + I$ is an adjacency matrix with self-loops, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ is a diagonal matrix with node degrees, and I is an identity matrix. We employ this convolution to build graph convolutional networks (GCNs) in our experiments. This formulation is a particular case of a more general approximate spectral graph convolution [8], in which case $\bar{X} \in \mathbb{R}^{N \times CK}$ are multiscale features averaged over K-hop neighbourhoods. Multiscale (multihop) graph convolution is used in our experiments with ChebyNet.

3 Multigraph Convolution

In graph convolution (Eq. 1), the graph adjacency matrix A encodes a single ($R = 1$) relation type between nodes. We extend Eq. 1 to a multigraph: a graph with multiple ($R \geq 1$) edges (relations) between the same nodes, represented as a set of adjacency matrices $\{A^{(r)}\}_1^R$. Previous work used concatenation- or decomposition-based schemes [7, 29] to fuse multiple relations. Instead, to capture more complex multirelational features, we adopt and extend recently proposed fusion methods from [20].

Two of the methods from [20] have certain limitations preventing us to adopt them directly in this work. In particular, the approach based on multidimensional Chebyshev polynomials is often infeasible to compute and was not shown to be superior in downstream tasks. In our experience, we also found that the multiplicative fusion was unstable to train. To that end, motivated by the success of multilayer projections in Graph Isomorphism Networks [37], we propose two simple yet powerful fusion methods (Figure 2).

Given features $\bar{X}^{(r)}$ corresponding to a relation type r, in the first approach we concatenate features for all relation types and then transform them using a two layer fully-connected network f with F_{hid} hidden units and the ReLU activation:

$$Y = f_{\text{CP}}([\bar{X}^{(0)}, \bar{X}^{(1)}, \ldots, \bar{X}^{(R-1)}]), \tag{2}$$

where the first and second layers of f_{CP} have $CKR \times F_{\text{hid}}$ and $F_{\text{hid}} \times F$ trainable parameters respectively, ignoring a bias. The second approach, illustrated in Figure 2, is similar to multiplicative/additive fusion [20], but instead of multiplication/addition we use concatenation:

$$Y = [\bar{Z}^{(0)}, \bar{Z}^{(1)}, ..., \bar{Z}^{(R-1)}]\Theta, \tag{3}$$

where $Z^{(r)} = f_r(\bar{X}^{(r)})$ and f_r is a single layer fully-connected network with F_{hid} output units followed by a nonlinearity, so that f_r and Θ have $CK \times F_{\text{hid}}$ and $RF_{\text{hid}} \times F$ trainable parameters respectively, ignoring a bias. Hereafter, we denote the first approach as CP (concatenation followed by projection) and the second as PC (projection followed by concatenation).

4 Multigraph Convolutional Networks

Image classification was recently formulated as a graph classification problem in [8, 11, 26], who considered small-scale image classification problems such as MNIST [23]. In this work, we present a model that scales to more complex and larger image datasets, such as PASCAL VOC 2012 [10]. We follow [26] and compute SLIC [1] superpixels for each image and build a graph, in which each node v_i corresponds to a superpixel and edges A_{ij} $(i, j \in [1,N])$ are computed based on the Euclidean distance between the coordinates $p_i \in \mathbb{R}^2$ and $p_j \in \mathbb{R}^2$ of their centres of masses using a Gaussian with some fixed width σ:

$$A_{ij}^{(r_{\text{spatial}})} = e^{\left(-\frac{||p_i - p_j||^2}{2\sigma^2}\right)}. \tag{4}$$

A frequent assumption of current GCNs is that there is at most one edge between any pair of nodes in a graph. This restriction is usually implied by datasets with such structure, so that in many datasets, graphs are annotated with the single most important relation type. Meanwhile, data is often complex and nodes tend to have multiple relationships of different semantic, physical, or abstract meanings. Therefore, we argue that there could be other relationships captured by relaxing this restriction and allowing for multiple kinds of edges, beyond those hardcoded in the data (*e.g.* spatial in Eq. 4).

4.1 Learning Flat vs. Hierarchical Edges

Prior work, *e.g.* [4, 29], proposed methods to learn from multiple edges, but similarly to the methods with a single edge type [19], they leveraged only predefined edges in the data. We formulate a more flexible model, which, in addition to learning from an arbitrary number of relations between nodes (see Section 3), learns abstract edges jointly with a GCN.

4.1.1 Flat Learnable Edges

We combine ideas from [3, 15, 31, 35] and propose to learn a new edge $A_{ij}^{(r_{\text{learn},l})}$ from any node v_i to node v_j with coordinates p_i and p_j using a trainable similarity function:

$$A_{ij}^{(r_{\text{learn},l})} = f_{\text{edge}}^{(l)}(p_i, p_j), j \in \mathcal{N}_i^{\varepsilon}, \tag{5}$$

where $l \in [1, L]$ indexes relation types; L is the number of learned relation types; and $\mathcal{N}_i^{\varepsilon}$ is a spatial neighbourhood of radius ε around node v_i. Using relatively small $\mathcal{N}_i^{\varepsilon}$ limits a model's predictive power, but is important for regularization and to meet computational requirements for larger images from which we extract many superpixels, such as in the PASCAL dataset. We also experimented with feeding node features, such as mean pixel intensities, to this function, but it did not give positive outcomes. Instead, we further regularize the model by constraining the input to be the absolute coordinate differences in lieu of raw coordinates and applying the softmax on top of the predictions:

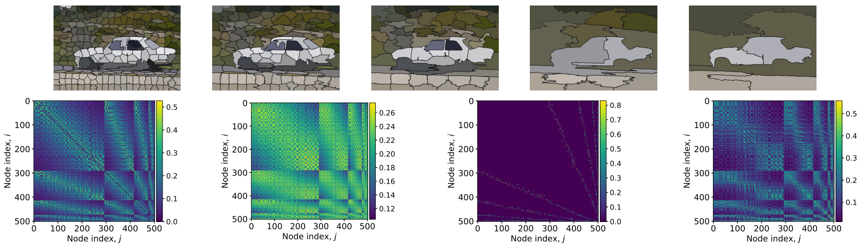
$$f^{(l)}_{\text{edge}}(p_i, p_j) = \text{softmax}\left[f^{(l)}(|p_i - p_j|)\right], j \in \mathcal{N}^{\varepsilon}_i, \tag{6}$$

where f is a small two layer fully-connected network with L output units and $f^{(l)}$ denotes taking the l-th output. Using the absolute value makes the filters symmetric (Figure 5), but still sensitive to orientation as opposed to the spatial edge defined in Eq. 4. The softmax is used to encourage sparse (more localized) edges.

Our approach to predict edges can be viewed as a particular case of generative models of graphs, such as [32]. However, the objectives of our model and the latter are orthogonal. Generative models typically make probabilistic predictions to induce diversity of generated graphs, and generation of each edge, node and their attributes should respect other (already generated) edges and nodes in the graph, so that the entire graph becomes plausible. In our work, we aim to scale our model to larger graphs and assume locality of relationships in visual data, therefore we design a 1) simple deterministic function 2) that makes predictions depending only on local information.

4.1.2 Hierarchical Edges

Depending on the number of SLIC superpixels, we can build different levels of image abstraction. Low levels of abstraction maintain fine-grained details, whereas higher levels mainly preserve global structure. To leverage useful information from multiple levels, we first compute superpixels at several different scales. Then, spatial and learnable relations are computed using Eq. 4 and 6, treating nodes at different scales as a joint set of superpixels. This creates a graph of multiscale image representation (Figure 3 *(a,b)*). However, this representation is still flat and, thus, limited.



(a) spatial ($K = 1$) *(b)* spatial ($K = 7$) *(c)* hierarchy ($K = 1$) (d) hierarchy ($K = 7$)

Figure 3: *(top)* We compute superpixels at several scales and combine all of them into a single set. *(bottom)* We then build a graph, where each node corresponds to a superpixel from this set and has features, such as mean RGB color and coordinates of the centres of masses. Using Eq. 4 and 7, we compute spatial *(a)* and hierarchical *(c)* edges. Nodes 0 to 300 correspond to the first level of the hierarchy (first scale of superpixels), and nodes 300 to 400 correspond to the second level, and so forth. Notice that spatial edges *(a)* are created both within and between levels, while hierarchical *(c)* edges exist only between hierarchical levels. *(c, d)* Powers of the adjacency matrices used in a multiscale ChebyNet allow information to diffuse over the graph making it possible to learn filters with more global support.

To alleviate this, we introduce a novel hierarchical graph model, where child-parent relations are based on intersection over union (IoU) between superpixels v_i and v_j at different scales:

$$A_{ij}^{(r_{\text{hier}})} = \text{IoU}(v_i, v_j), \tag{7}$$

where $A^{(r_{\text{hier}})}$ is an adjacency matrix of hierarchical edges and $A_{ij}^{(r_{\text{hier}})} = 0$ for nodes at the same scale (Figure 3 *(c,d)*). Using IoU means that child nodes can have multiple parents. To guarantee a single parent, hierarchical superpixel algorithms can be considered [2].

4.2 Layer vs. Global pooling

Inspired by convolutional networks, previous works [6, 8, 11, 26, 31, 39] built an analogy of pooling layers in graphs, for example, using the Graclus clustering algorithm [9]. In CNNs, pooling is an effective way to reduce memory and computation, particularly for large inputs. It also provides additional robustness to local deformations and leads to faster growth of receptive fields. However, we can build a convolutional network without any pooling layers with similar performance on a downstream task [33] — it just will be relatively slow, since pooling is extremely cheap on regular grids, such as images. In graph classification tasks, the input dimensionality, which corresponds to the number of nodes $N = |\mathcal{V}|$, is often very small ($\sim 10^2$) and the benefits of pooling are less clear. Graph pooling, such as in [9], is also computationally intensive since we need to run the clustering algorithm for each graph independently, which limits the scale of problems we can address. Aiming to simplify the model while maintaining classification accuracy, we exclude pooling layers between graph convolutional layers and perform global maximum pooling over nodes following the last conv. layer. This fixes the size of the penultimate feature vector irrespective of N (Figure 4).

5 Experiments

We evaluate our model on three image-based graph classification datasets: MNIST [23], CIFAR-10 [22] and PASCAL Visual Object Classes 2012 (PASCAL) [10]. For each dataset, there is a set of graphs with different numbers of nodes (superpixels), and each graph $\mathcal{G}$ has a single categorical label that is to be predicted. For baselines we use a single ($R = 1$) spatial relation type defined using Eq. 4. For PASCAL we predict multiple categorical labels per image and report mean average precision (mAP).

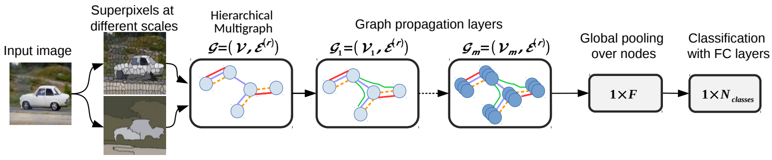

Input image
Superpixels at different scales
Hierarchical Multigraph
$\mathcal{G}=(\mathcal{V},\mathcal{E}^{(r)})$
Graph propagation layers
$\mathcal{G}_1=(\mathcal{V}_1,\mathcal{E}^{(r)})$
$\mathcal{G}_m=(\mathcal{V}_m,\mathcal{E}^{(r)})$
Global pooling over nodes
$1\times F$
Classification with FC layers
$1\times N_{classes}$

Figure 4: Image classification pipeline using our model. Each m^{th} graph convolutional layer in our model takes the graph $\mathcal{G}_m = (\mathcal{V}_m, \mathcal{E}^{(r)})$ and returns a graph with the same nodes and edges. Node features become increasingly global after each subsequent layer as the receptive field increases, while edges are propagated without changes. As a result, after several graph convolutional layers, each node in the graph contains information about its immediate neighbours and a large neighbourhood around it. By pooling over nodes we summarize the information collected by each node. Fully-connected layers follow global pooling to perform classification.

MNIST consists of 70k greyscale 28×28px images of handwritten digits. CIFAR-10 has 60k coloured 32×32px images of 10 categories of animals and vehicles. PASCAL is a more challenging dataset with realistic high resolution images (typically around 300×500px) of 20 object categories (people, animals, vehicles, indoor objects). We use standard classification splits, training our model on 5,717 images and reporting results on 5,823 validation images. We note that CIFAR-10 and PASCAL have not been previously considered for graph-based image classification, and in this work we scale our method to these datasets. In fact, during experimentation we found some other graph convolutional methods unable to scale (see Section 5.3).

5.1 Architectural and Experimental Details

GCNs. In all experiments, we train GCNs, ChebyNets and MoNet [26] with three graph convolutional layers, having 32, 64 and 512 filters with the ReLU activation after each layer followed by global max pooling and a fully-connected classification layer (Figure 4). For MNIST, we use a dropout rate [34] of 0.5 before the class. layer, while for CIFAR-10 and PASCAL instead we employ batch normalization (BN) [16] after each convolutional layer. For edge fusion, projection f_{CP} in Eq. 2 is modelled by a two layer network with $F_{\text{hid}} = 64$ hidden units and the ReLU activation between layers. Similarly, projections $f_r(\bar{X}^{(r)})$ in Eq. 3 are single layer neural networks with $F_{\text{hid}} = 64$ output units followed by the tanh activation. The edge prediction function in f_{edge} (Eq. 6) is a two layer neural network with 32 hidden units, L output units, and neighbourhood $\mathcal{N}_i^{\varepsilon}$ is set to the $0.2N$ spatially nearest nodes.

ConvNets (CNNs). To allow fair comparison, we train CNNs with the same number of filters in convolutional layers as GCNs, with filters of size 3 for MNIST, 5 for CIFAR-10 and 7 for PASCAL, max pooling between layers and global average pooling after the last convolutional layer. Deeper and larger CNNs and GCNs can be trained to further improve results. Since images fed to CNNs are defined on a regular grid, adding coordinate features is uninformative, because these features are exactly the same for all examples in the dataset.

Low resolution ConvNets. GCNs take a relatively low resolution input compared to CNNs: 75 vs. 784 for MNIST, 150 vs. 1024 for CIFAR-10 and 1000 vs. 150,000 for PASCAL. Factors by which it is reduced are 10.5, 6.8 and 150 respectively. Therefore, direct comparison of GCNs to CNNs is unfair. To provide an alternative (yet not perfect) form of comparison, we design experiments with low resolution inputs fed to CNNs. In particular, to match the spatial dimensions of inputs to GCNs and CNNs, for MNIST we reduce the size to 9×9 px, for CIFAR-10 to 12×12 px and for PASCAL to 32×32 px taking into account that the average number of superpixels returned by the SLIC algorithm is often smaller than requested. Admittedly, downsampling using SLIC superpixels is more structural than bilinear downsampling, so GCNs receive a stronger signal, but we believe it is still an interesting experiment. Principal component analysis could be used as a more adequate way to implicitly reduce the input dimensionality for CNNs, but this method is infeasible for large images, such as in PASCAL, while superpixels can be easily computed in such cases. For comparison, we also report results on low resolution images using GCNs.

Training. We train all models using Adam [18] with learning rate of 1e$-$3, weight decay of 1e$-$4, and batch size of 32. For MNIST, the learning rate is decayed after 20 and 25 epochs and models are trained for 30 epochs. For CIFAR-10 and PASCAL, the learning rate is decayed after 35 and 45 epochs and models are trained for 50 epochs. We train models using four edge fusion methods: concatenation-based baseline, additive fusion [20] and two methods (CP and PC) introduced in Section 3 (see Table 1 and Figure 6 *(a)* for a summary).

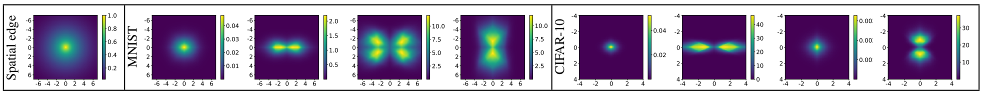


Figure 5: Examples of predicted edges for MNIST and CIFAR-10 using the models with $L = 4$ learned edges (Eq. 6). These predictions can be interpreted as filters centred at some node v_i, so that values along both axes denote distances from node v_i to other nodes v_j and the intensity denotes the strength of a connection with that node. As we can see, our models learn filters with variable intensity depending on the direction to better capture features in images. For comparison, a spatial edge (Eq. 4) is shown on the left, which has the same value across all directions limiting model's flexibility.

In each case, we train a model 5 times with different random seeds and report average results and standard deviation. In Table 2, we report the best results over different fusion methods superscripting the best fusion method.

Graph formation for images. For MNIST, we compute a hierarchy of 75, 21, and 7 SLIC [1] superpixels and use mean superpixel intensity and coordinates as node features, so $X \in \mathbb{R}^{N\times3}$. For CIFAR-10, we add one more level of 150 superpixels and also use coordinate features, so $X \in \mathbb{R}^{N\times5}$. For PASCAL, due to its more challenging images, we further add two more levels of 1,000 and 300 superpixels, so $X \in \mathbb{R}^{N\times5}$. Note, the number of superpixels provided above are upper bounds we impose on the SLIC algorithm. For low resolution experiments with GCNs, we build graphs from pixels and their coordinates, that is the graph structure is the same across all examples in the dataset.

5.2 Results

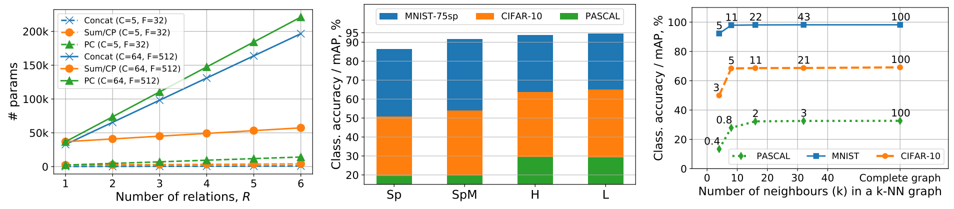
The image classification results on the MNIST, CIFAR-10 and PASCAL datasets are presented in Table 2. We first observe that among GCN baselines, MoNet [26] and ChebyNet [8] show comparable performance, significantly outperforming GCN [19], which can be explained by very local filters in the latter. Next, the hierarchical (H) and learned (L or L4, i.e. models with $L = 4$ learnable edges) connections proposed in this work are shown to be complementary, substantially improving both GCNs and ChebyNets with a single spatial edge type. Additional qualitative and quantitative analysis is provide in Table 1, Figures 5 and 6. By combining hierarchical and learned edge types (H-L, H-L4) and adding multiscale filters, we achieve a further gain in accuracy while keeping the number of trainable parameters low compared to ConvNets, MoNet and ChebyNets with large K. Importantly, our multirelational GCNs also show better results than ConvNets with a low resolution input.

In Table 2, for ChebyNet we report results using the best K chosen from the range [2, 20]: $K = 10, 3, 2$ for MNIST, CIFAR-10 and PASCAL respectively in case of a baseline ChebyNet and $K = 4, 4, 2$ in case of H-L-ChebyNet. Among evaluated fusion methods, CP and additive (S) work best for MNIST, whereas PC and CP dominate for CIFAR-10 and PASCAL.

We also evaluate a baseline GCN on low resolution images to highlight the importance of SLIC superpixels compared to downsampled pixels (Table 2). Surprisingly, SLIC features provide only a moderate improvement compared to pixels bringing us to two conclusions. First, average intensity values of superpixels and their coordinates are rather weak features that carry limited geometry and texture information. Stronger features can be boundaries of superpixels, but efficient and effective modeling of such information remains an open question. Second, we hypothesize that on full resolution images GCNs can potentially reach the performance of ConvNets or even outperform them, while having appealing properties, such as in [17]. However, to scale GCNs to such large graphs, two approaches should be

	Fusion method		# params	$K=1$ (GCN)	$K=2$	$K=7$
$R=1$	(Sp)	Spatial edge (Eq. 1,4)		86.36±0.95	97.08±0.11	98.17±0.02
	(SpM)	Spatial multiscale (Eq. 1,4)	$C \times K \times F$	91.74±0.28	97.38±0.09	98.10±0.05
	(H)	Hier. edge (Eq. 1,7)		93.65±0.07	96.94±0.07	97.07±0.07
$R=2$	(C)	Concat [20]	$C \times K \times R \times F$	97.07±0.06	97.95±0.07	98.28±0.11
	(S)	Sum [20]	$F_{\text{hid}}(C \times K \times R + F)$	**97.93**±0.05	98.30±0.01	98.44±0.10
	(CP)	C-proj (Eq. 2)	$F_{\text{hid}}(C \times K \times R + F)$	97.77±0.08	**98.31±0.09**	**98.52±0.09**
	(PC)	Proj-c (Eq. 3)	$RF_{\text{hid}}(C \times K + F)$	97.87±0.04	98.28±0.04	98.47±0.09
	Max gain			4.28	0.93	0.35

Table 1: Accuracy gain achieved on MNIST-75sp (MNIST with 75 superpixels) by using both spatial and hierarchical edges ($R = 2$) versus a single edge ($R = 1$) depending on edge fusion methods for different number of hops, K, of filters.



(a) *(b)* *(c)*

Figure 6: *(a)* Number of trainable parameters, # params, in a graph convolutional layer as a function of the number of relations, R. Fusion methods based on trainable projections, including those proposed in our work, have "# params" comparable to the baseline concatenation method while being more powerful in terms of classification (see Tables 1 and 2). *(b)* Comparison of single edge types, where learned and hierarchical edges outperform spatial edges. *(c)* Effect of graph sparsification on classification accuracy. Numeric labels over markers denote sparsity of a graph (%). Notice little or no decrease in accuracy for sparse graphs, even if they have only 2-10% non-zero edges.

considered: 1) fast and effective pooling methods [12, 39]; 2) sparsification of graphs, which we evaluated and compared to complete graphs used in our experiments (Figure 6 *(c)*).

5.3 Discussion

Our method relies on Graph Convolutional Networks (GCN) [19] and its multiscale variant ChebyNet [8]). While ChebyNet is superior to GCN in case of a single relation type due to a larger receptive field size, adding multiple relations make both methods comparable. In fact, we show that adding hierarchical edges is generally more advantageous than adding multihop ones, because hierarchy is a strong inductive bias that facilitates capturing features of spatially remote, but hierarchically close nodes (Table 1). Learned edges also improve on spatial ones, which are defined heuristically in Eq. 6 and therefore might be suboptimal.

Closely related to our work, [26] formulated the generalized graph convolution model (MoNet) based on a trainable transformation to pseudo-coordinates, giving rise to anisotropic kernels and excellent results in visual tasks. However, we found our models with multiple relations to be better (Table 2). Notably, the computational cost (both memory and speed) of MoNet is higher than for any of our models due to the costly patch operator in [26], so we could not perform experiments on PASCAL with 1000 superpixels due to limited GPU memory. The argument previously made in favour of MoNet against spectral methods, including ChebyNet, was the sensitivity of spectral convolution methods to changes in graph size and structure. We contradict this argument and show strong performance of ChebyNet.

	Model	R	**MNIST**	**CIFAR-10**	**PASCAL**	**# params**
Input	ConvNets, full res, N_{pixels}		784	1024	150 000	
	ConvNets/GCNs, low res, N_{pixels}		81	144	1024	
	GCNs, $N_{\text{SLIC\_superpixels}}$		$\leq$75	$\leq$150	$\leq$1000	
Pixels: baselines	ConvNet [23], full res	–	99.42±0.01	83.77±0.21	41.42±0.51	320-360k
	ConvNet [23], low res	–	97.09±0.10	72.68±0.40	32.69±0.43	320-330k
	GCN [19], low res	1	81.36±0.41	50.57±0.14	18.57±0.10	42k
SLIC: baselines	GCN [19]	1	86.29±0.44	51.51±0.55	19.24±0.06	42k
	MoNet [26]	1	96.64±2.01	72.62±0.57	‡	880k
	ChebyNet [8]	1	98.24±0.03	68.92±0.23	32.38±0.38	80-350k
SLIC: ours	L-GCN	2	97.64±0.12^{S} (↑1.05)	70.14±0.29^{PC} (↑1.00)	32.39±0.19^{PC} (↑0.08)	100k
	H-GCN	2	97.93±0.05^{S} (↑0.86)	69.05±0.14^{PC} (↑0.60)	32.18±0.29^{PC} (↑0.53)	100k
	H-L-GCN	3	98.35±0.09^{S} (↑0.63)	71.44±0.30^{CP} (↑1.81)	31.75±0.74^{PC} (↑0.54)	145k
	L4-GCN	5	98.42±0.12^{CP} (↑0.10)	72.67±0.36^{S,C} (0.00)	33.01±0.49^{C} (↓0.38^{PC})	185k
	H-L4-GCN	6	98.66±0.03^{S} (↑0.22)	72.89±0.70^{CP} (↑0.89)	32.61±0.45^{C} (↓2.18^{S})	220k
	H-L-ChebyNet	3	**98.68**±0.05^{CP} (↑0.18)	**73.18±0.52**^{PC} (↑2.40)	**34.46±0.47**^{PC} (↑1.46)	200k

Table 2: Image classification results: accuracy for MNIST and CIFAR-10 and mAP for PASCAL (%). Superscripts denote the best fusion method among the four studied methods (See Table 1 for notation). ‡ - unable to evaluate on our infrastructure due to high computational cost. # params is the total number of trainable parameters in a model (largest across rows). An up-arrow (↑) shows the gain compared to the baseline concatenation fusion; a down-arrow (↓) shows the loss of accuracy.

Another method, SplineCNN [11], is similar to MoNet and is also based on pseudo-coordinates, but we leave studying this method for future work. Note that performance of MoNet and SplineCNN on general graph classification problems where coordinates are not well defined is inferior compared to ChebyNet [20].

Finally, a family of methods based on graph kernels [21, 30] shows strong results on some non-visual graph classification datasets, but their application is limited to small scale graph problems with discrete node features, whereas we have real-valued features. Scalable extensions of kernel methods to graphs with continuous features were proposed [27, 38], but they still tend to be less competitive than methods based on GCN and ChebyNet [20].

6 Conclusion

We address several limitations of current graph convolutional networks and show improved graph classification results on a number of image datasets. First, we formulate the classification problem in terms of multigraphs, and extend edge fusion methods based on trainable projections. Second, we propose hierarchical edges and a way to learn new edges in a graph jointly with a graph classification model. Our results show that spatial, hierarchical, learned and multihop edges have a complimentary nature, improving accuracy when combined. We show that our models can outperform standard convolutional networks in experiments with low resolution images, which should pave the way for future research in that direction.

Acknowledgments

This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA). The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. The authors also acknowledge support from the Canadian Institute for Advanced Research and the Canada Foundation for Innovation.

References

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282, 2012.

[2] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):898–916, 2010.

[3] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[4] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795, 2013.

[5] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4): 18–42, 2017.

[6] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representations (ICLR)*, 2014.

[7] Xinlei Chen, Li-Jia Li, Li Fei-Fei, and Abhinav Gupta. Iterative visual reasoning beyond convolutions. In *Proc. CVPR*, 2018.

[8] 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.

[9] Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. Weighted graph cuts without eigenvectors a multilevel approach. *IEEE transactions on pattern analysis and machine intelligence*, 29(11), 2007.

[10] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2): 303–338, 2010.

[11] Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. Splinecnn: Fast geometric deep learning with continuous b-spline kernels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 869–877, 2018.

[12] Hongyang Gao and Shuiwang Ji. Graph U-Net. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019.

[13] 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 (ICML)*, pages 1263–1272, 2017.

[14] William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.

[15] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

[16] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015.

[17] Renata Khasanova and Pascal Frossard. Graph-based isometry invariant representation learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1847–1856. JMLR. org, 2017.

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[19] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[20] Boris Knyazev, Xiao Lin, Mohamed R 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.

[21] Nils M Kriege, Pierre-Louis Giscard, and Richard Wilson. On valid optimal assignment kernels and applications to graph classification. In *Advances in Neural Information Processing Systems*, pages 1623–1631, 2016.

[22] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[23] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[24] Xiaodan Liang, Xiaohui Shen, Jiashi Feng, Liang Lin, and Shuicheng Yan. Semantic object parsing with graph lstm. In *European Conference on Computer Vision*, pages 125–143. Springer, 2016.

[25] Cewu Lu, Ranjay Krishna, Michael Bernstein, and Li Fei-Fei. Visual relationship detection with language priors. In *European Conference on Computer Vision*, 2016.

[26] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proc. CVPR*, volume 1, page 3, 2017.

[27] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 2014–2023, 2016.

[28] Nikita Prabhu and R Venkatesh Babu. Attribute-graph: A graph based approach to image ranking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1071–1079, 2015.

[29] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer, 2018.

[30] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(Sep): 2539–2561, 2011.

[31] Martin Simonovsky and Nikos Komodakis. Dynamic edgeconditioned filters in convolutional neural networks on graphs. In *Proc. CVPR*, 2017.

[32] Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In *International Conference on Artificial Neural Networks*, pages 412–422. Springer, 2018.

[33] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[34] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[35] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[36] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. In *International Conference on Learning Representations (ICLR)*, 2019.

[37] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations (ICLR)*, 2019.

[38] Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1365–1374. ACM, 2015.

[39] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems*, pages 4805–4815, 2018.